

**T.C.
MİLLÎ EĞİTİM BAKANLIĞI**

BİLİŞİM TEKNOLOJİLERİ

GELİŞMİŞ İNTERNET UYGULAMALARINDA KONTROLLER VE VERİ BAĞLAMA

Ankara, 2016

- Bu modül, mesleki ve teknik eğitim okul / kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

İÇİNDEKİLER.....	i
AÇIKLAMALAR	iii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	2
1. TEMEL KONTROLLER.....	2
1.1. Metin Kutusu(TextBox).....	2
1.2. Takvim(Calendar)	5
1.3. Tarih Seçici(DatePicker).....	10
1.4.Sürgü(Slider).....	14
UYGULAMA FAALİYETİ	19
ÖLÇME VE DEĞERLENDİRME	23
ÖĞRENME FAALİYETİ-2	25
2. İÇERİK KONTROLLERİ.....	25
2.1.Düğme(Button)	25
2.2. Onay Kutusu(CheckBox).....	28
2.3. Radyo Düğmesi(RadioButton).....	35
2.4. Popup	38
2.5. ScrollViewer	40
UYGULAMA FAALİYETİ	42
UYGULAMA FAALİYETİ-2	45
ÖLÇME VE DEĞERLENDİRME	47
ÖĞRENME FAALİYETİ.....	49
3. GELİŞMİŞ KONTROLLER.....	49
3.1.ItemsControl Kullanımı	49
3.2. DataGrid Kullanımı.....	53
3.3. ListBox Kullanımı	60
UYGULAMA FAALİYETİ	69
ÖLÇME VE DEĞERLENDİRME	74
ÖĞRENME FAALİYETİ-4	76
4. ÇOKLU ORTAM KONTROLLERİ.....	76
4.1.MediaElement Kontrolü.....	76
4.2.MediaElement Kontrolünü Programlama	78
4.3.VideoBrush	81
UYGULAMA FAALİYETİ	83
ÖLÇME VE DEĞERLENDİRME	87
ÖĞRENME FAALİYETİ-5	89
2. BAĞLAMA İŞLEMLERİ.....	89
5.1.Bağlama Kuralları	89
5.2. Veri Bağlama(DataContext)	94
5.3.Liste Bağlama(List Binding).....	96
5.4.Ver Şablonu(DataTemplate)	100
5.5. Master/Detail Data Binding	104

UYGULAMA FAALİYETİ	110
ÖLÇME VE DEĞERLENDİRME	114
MODÜL DEĞERLENDİRME	115
CEVAP ANAHTARLARI	118

AÇIKLAMALAR

ALAN	Bilişim Teknolojileri
DAL/MESLEK	Veri Tabanı Programcılığı
MODÜLÜN ADI	Gelişmiş İnternet Uygulamalarında Kontroller ve Veri Bağlama
MODÜLÜN TANIMI	Bu modül, XAML tabanlı internet uygulamaları geliştirmede kullanılan bazı kontroller ve veri bağlama işlemleri hakkında gerekli bilgileri edindirmek için hazırlanan bir eğitim materyalidir.
SÜRE	40/32
ÖN KOŞUL	
YETERLİK	Kontrolleri kullanabilme ve veri bağlama işlemlerini gerçekleştirebilmek
MODÜLÜN AMACI	Genel Amaç Bu modül ile gerekli ortam sağlandığında; kontrolleri kullanabilecek ve veri bağlama işlemlerini gerçekleştirebileceksiniz. Amaçlar 1. Temel kontrolleri kullanabileceksiniz. 2. İçerik kontrollerini kullanabileceksiniz. 3. Gelişmiş kontrolleri kullanabileceksiniz. 4. Çoklu ortam kontrollerini kullanabileceksiniz. 5. Veri bağlama işlemlerini gerçekleştirebileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANİMLÂRI	Ortam: İnternete ve ağı bağlı bilgisayar laboratuvarı Donanım: Bilgisayar, Tasarım ve Programlama Yazılımları
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen, modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Son yıllarda yaygın olarak kullanılan XAML tabanlı yazılım geliştirme araçları, internet uygulamaları geliştirmede önemli kolaylıkları ve yenilikleri beraberinde getirmiştir. XAML teknolojisi sayesinde veritabanı alt yapısına sahip etkileşimli internet uygulamaları geliştirmek oldukça kolay bir hâle dönüşmüştür.

Faydalanacağınız bu modül, XAML tabanlı internet uygulamaları geliştirmede kullanılan bir takım kontroller hakkında yeterli düzeyde bilgiler sağlamaktadır. Ayrıca modül içerisinde, kontrollerin birbiriyle ya da veri kaynakları ile veri alışverişi sağlamasında kullanılan teknikler hakkında gerekli bilgilere de yer verilmiştir.

XAML teknolojisi ile internet uygulamalarını geliştirmede büyük bir adım daha atmanızı sağlayacaktır.

ÖĞRENME FAALİYETİ-1

AMAÇ

Temel kontrolleri kullanabileceksiniz.

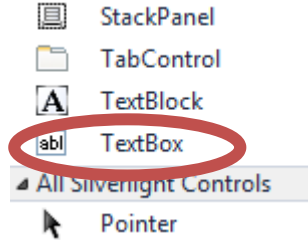
ARAŞTIRMA

- Metin kutusu(TextBox),takvim(Calendar), tarih seçici(DatePicker)ve sürgü(Slider) kontrollerini araştırınız.

1. TEMEL KONTROLLER

1.1. Metin Kutusu(TextBox)

Sayfa üzerinden bilgi girişi yapmayı sağlayan kontroldür. Araç kutusu(Toolbox) kullanılarak veya XAML kodları ile sayfaya eklenebilir.



Resim 1.1: ToolBox üzerinde TextBox kontrolü

Sayfaya metin kutusunu(TextBox) ekledikten sonra “Properties” panelini kullanarak ya da XAML kodlarını düzenleyerek bu kontrolün özelliklerini değiştirebiliriz.

“TextBox” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
Text	“TextBox”içerisine girilen metni temsil eder.	“String” tipte veri tutar.
IsEnabled	“TextBox” kontrolünü aktif ya da pasif yapmak için kullanılır.	Aktif olması için “True”, pasif olması için ise “False” değerini almalıdır.
MaxLength	Girilebilecek maksimum karakter sayısını belirler.	Tamsayı(int) türünde değer alır.
FontFamily	Girilen metnin yazı tipini belirler.	
FontWeight	Girilen metnin kalın olup olmayacağını belirler.	Metnin kalın olması için “Bold”, normal olması için “Normal” değerini almalıdır.
FontStyle	Girilen metnin eğik olup olmayacağını belirler.	Metnin eğik olması için “Italic”, düz olması için “Normal” değerini almalıdır.
FontSize	Yazı büyüklüğünü değiştirir.	Sayısal değer alır.
TextWrapping	Girilen metnin “TextBox” genişliğinden fazla olması durumunda alt satıra geçilip geçilmeyeceğini belirler.	Alt satıra geçmesi için “Wrap”, geçmemesi için “NoWrap” değerini alır.
Background	Zemin rengini değiştirir.	
BorderBrush	Çerçeve rengini değiştirir.	
Foreground	Yazı rengini değiştirir.	
BorderThickness	Çerçeve kalınlığını değiştirir.	

Tablo 1.1: “TextBox” özellikleri

“TextBox” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

Örnek:

```
<Grid x:Name="LayoutRoot" Background="White">

<TextBox Background="Red" Height="30" Name="textBox1" Width="180"
Foreground="White" FontWeight="Bold" BorderThickness="3"
FontFamily="Comic Sans MS" FontSize="12" />

</Grid>
```

➤ Çalışma anı görüntüsü



Resim 1.2: Çalışma anı görüntüsü

“TextBox” kontrolüne ait en önemli olay “**TextChanged**” olayıdır. Uygulamanın çalışması sırasında metin kutusu içeriği her değiştiğinde “TextChanged” olayı meydana gelir.

Örnek: Veri girilirken kendi içeriğini kontrol eden ve girilen bilginin doğru ya da yanlış olmasına göre zemin rengini yeşil ya da kırmızı yapan bir “TextBox” örneğini inceleyelim.

➤ Tasarım görüntüsü



Resim 1.3: Tasarım görüntüsü

Program Kodları: “TextBox”un “TextChanged” olayına yazılan kodlar:

```
private void textBox1_TextChanged(object sender, TextChangedEventArgs e)
{
    SolidColorBrush renk = new SolidColorBrush();
    if (textBox1.Text == "ISPARTA-KTML")
    {
        renk.Color = Colors.Green;
    }
}
```

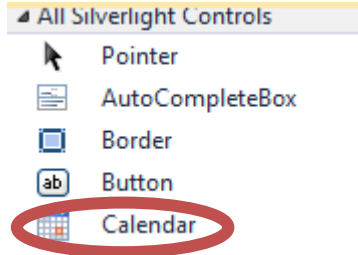
➤ Çalışma anı görüntüsü



Resim 1.4: Metin kutusu çalışması

1.2. Takvim(Calendar)

Üzerinden tarih seçimi yapmayı sağlayan kontroldür. Araç kutusu (Toolbox)ya da XAML kodları kullanarak sayfaya takvim(Calendar) kontrolünü ekleyebiliriz.



Resim 1.5: ToolBox üzerinde Calendar kontrolü

“Calendar” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
SelectedDate	Seçili olan tarihi almak ya da bir tarihi seçili yapmak için kullanılır.	“ DateTime ” türünde veri tutar.
DisplayDate	İstenen tarihi göstermek için kullanılır.	“ DateTime ” türünde veri tutar.
DisplayDateStart	Gösterilmek istenilen tarih aralığının başlangıç tarihini belirler.	“ DateTime ” türünde veri tutar.
DisplayDateEnd	Gösterilmek istenilen tarih aralığının bitiş tarihini belirler.	“ DateTime ” türünde veri tutar.
DisplayMode	Yılları, yılın aylarını veya ayın günlerini göstermek için kullanılır.	Alabileceği değerler: Decade : Yılları gösterir. Year : Yıldaki ayları gösterir. Month : Aydaki günleri gösterir.
SelectionMode	Tek bir tarih seçebilme, bir tarih aralığı seçebilme, birden fazla tarih aralığı seçebilme ya da tarih seçimine kapatma seçeneklerini sunar.	Alabileceği değerler: None : Tarih seçimi yapılamaz. SingleDate : Tek bir tarih seçilebilir. SingleRange : Tek bir tarih aralığı seçilebilir. MultipleRange : Birden fazla tarih aralığı seçilebilir.
SelectedDates	Seçili olan tarihlere yeni bir tarih veya tarih aralığını ilave etmek için kullanılır.	“SelectedDates” sınıfının “Add()” ve “AddRange()” gibi metodları kullanılabilir.

FirstDayOfWeek	Haftanın ilk gününü belirlemek için kullanılır.	“DayOfWeek” sınıfında veri tutar.
IsTodayHighLighted	İçinde bulunduğumuz günün renkli olarak vurgulanmasını ya da vurgulanmamasını belirler.	“True” değerini aldığıda bugün vurgulanır. “False” değerini aldığıda vurgulanmaz.
BlackoutDates	Bazı tarih veya tarih aralıklarının seçilemez duruma getirilmesi için kullanılır.	Programlama dili kodları içinde“BlackoutDates” sınıfının“Add()” ve “AddDatesInPast()” gibi metotları kullanarak takvim üzerinde seçilemeyecek tarihler belirlenir.

Tablo 1.2: “Calendar” özellikleri

“Calendar” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

Örnek:XAML kodları ile sayfa üzerinde takvim kontrolü oluşturma

```
<Grid x:Name="LayoutRoot" Background="White">
<controls:Calendar Name="calendar1" DisplayDate="7/1/2013"
IsTodayHighlighted="True" SelectionMode="MultipleRange"
FirstDayOfWeek="Monday" />
```

➤ Çalışma anı görüntüsü



Resim 1.6:Takvim nesnesi

Örnek: Sayfa açılışında takvim özelliklerinin dinamik olarak değiştirilmesi.(5, 8 ve 10 Temmuz tarihlerini seçili olarak gösterecektir.)

```

public MainPage()
{
    InitializeComponent();

    DateTime tarih = new DateTime();

    calendar1.IsTodayHighlighted = false;

    calendar1.DisplayMode = CalendarMode.Month;

    calendar1.SelectionMode = CalendarSelectionMode.MultipleRange;

    tarih = DateTime.Parse("5/7/2013");
    calendar1.SelectedDates.Add(tarih);

    tarih = DateTime.Parse("8/7/2013");
    calendar1.SelectedDates.Add(tarih);

    tarih = DateTime.Parse("10/7/2013");
    calendar1.SelectedDates.Add(tarih);
}

```

➤ Çalışma anı görüntüsü

Temmuz 2013						
Pt	Sa	Ça	Pe	Cu	Ct	Pz
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Resim 1.7: Takvim üzerinde birden fazla tarih seçimi

“Calendar” kontrolüne ait en önemli olaylar ise şunlardır:

- **SelectedDatesChanged olayı:**“Calendar” üzerinde seçili olan bir tarih değiştirildiği anda çalışır.
- **DisplayDateChanged olayı:**“Calendar” üzerinde görüntülenen tarih değiştirildiği anda çalışır.
- **DisplayModeChanged olayı:** “Calendar”görüntü modları(“Decade”, “Year” ya da “Month”) arasında geçiş yapıldığı anda çalışır.

Örnek: Üzerinden seçilen tarihin günü “Cumartesi” ya da “Pazar” ise “Seçtiğiniz tarihe randevu verilemez!” mesajını, mesai günlerinden birisi ise “Randevu talebiniz kabul edilmiştir.” mesajını görüntüleyen bir “Calendar” örneğini inceleyelim.

- **Tasarım görüntüsü**

Temmuz 2013						
Pt	Sa	Ça	Pe	Cu	Ct	Pz
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

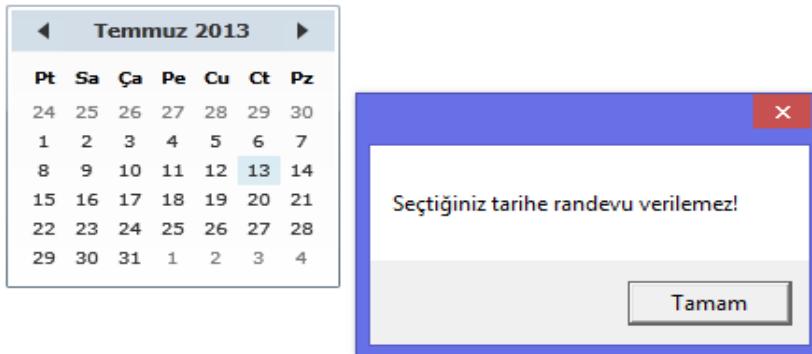
Resim 1.8: Takvim görüntüsü

Program Kodları:“SelectedDatesChanged” olayına yazılan kodlar:

```
private void calendar1_SelectedDatesChanged(object sender,
SelectionChangedEventArgs e)
{
    string gun = calendar1.SelectedDate.Value.DayOfWeek.ToString();
    if ( gun == "Saturday" || gun == "Sunday")
    {
        MessageBox.Show("Seçtiğiniz tarihe randevu verilemez!");
    }
    else
    {
        MessageBox.Show("Randevu talebiniz kabul edilmiştir.");
    }
}
```

Not:“Calendar” kontrolünün “SelectedDate” özelliğinin altında bulunan “Value” özelliği içinden yıl(Year), ay(Month), gün(Day) ve haftanın günü(DayOfWeek) gibi verilere ayrı ayrı ulaşabiliriz.

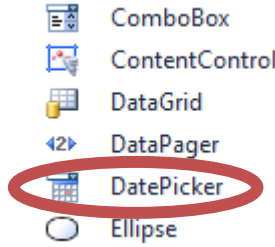
➤ **Ekran görüntüsü**



Resim 1.9: Uygulama sonucu

1.3. Tarih Seçici(DatePicker)

Açılıp kapanabilen bir takvime sahip metin kutusu olarak tanımlayabiliriz. Oldukça kullanışlı olan tarih seçici(DatePicker) kontrolü, takvimden seçilen tarihi metin kutusunda gösterebildiği gibi metin kutusuna girilen tarihi de takvimde seçili olarak gösterebilir.



Resim 1.10: ToolBox üzerinde DatePicker kontrolü

“DatePicker”’in takvimi bir önceki başlıkta ele alınan “Calendar” kontrolünün bütün özelliklerine sahiptir. Ancak “DatePicker”, farklı olarak “IsDropDownOpen” özelliğine de sahiptir. “IsDropDownOpen” özelliği “DatePicker”’deki takvimi açıp kapatan özelliktir. Takviminin açılması için “True”, kapanması için “False” değerini almalıdır.

“DatePicker” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

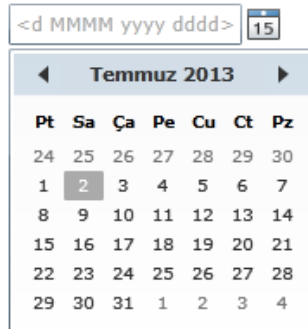
Örnek:

```
<Grid x:Name="LayoutRoot" Background="White">

<controls:DatePicker HorizontalAlignment="Left" Margin="44,76,0,0"
Name="datePicker1" VerticalAlignment="Top" SelectedDateFormat="Long"
IsDropDownOpen="True"/>

</Grid>
```

➤ Çalışma anı görüntüsü



Resim 1.11: Çalışma anı görüntüsü

Örnek: Sayfa açılışında “DatePicker” in takviminde “8-15 Temmuz 2013” tarihleri arasını seçilemez olarak işaretleyen bir örneği inceleyelim.

```
public MainPage()
{
    InitializeComponent();

    datePicker1.IsDropDownOpen = true;

    datePicker1.IsTodayHighlighted = false;

    CalendarDateRange aralik =
    newCalendarDateRange(DateTime.Parse("8/7/2013"),
    DateTime.Parse("15/7/2013"));

    datePicker1.BlackoutDates.Add(aralik);
}
```

Yukarıda sarı renkle vurgulanmış program kodu “CalendarDateRange” sınıfından “aralik” isminde bir nesne üretir. “aralik” nesnesini üretirken, “8-15 Temmuz 2013” tarih aralığının başlangıç ve bitiş tarihlerini, “DateTime” türünde iki parametre olarak almaktadır. Ayrıca oluşturulan tarih aralığı, BlackoutDates özelliğinin Add() metodu kullanılarak seçilemez tarihler listesine eklenmiş olur.

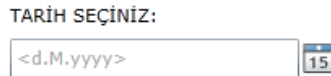


Resim 1.12: BlackoutDates özelliğinin kullanımı

“DatePicker” kontrolüne ait en önemli olaylar ise aşağıda belirtilmiştir.

- **SelectedDateChanged** olayı: “DatePicker” takviminde bir tarih seçildiği anda çalışır.
- **CalendarClosed** olayı: “DatePicker” takvimi kapatıldığı anda çalışır.
- **CalendarOpened** olayı: “DatePicker” takvimi açıldığı anda çalışır.

Örnek: Üzerinden seçilen tarihin okullara kayıt süreci(15 -31 Ağustos) içinde olup olmadığını bir mesaj kutusu yardımıyla bildiren “DatePicker” örneğini inceleyelim.



Resim 1.13: DatePicker

Program Kodları:“DatePicker”in “SelectedDateChanged” olayına yazılan kodlar:

```
privatevoid datePicker1_SelectedDateChanged_1(object sender,
SelectionChangedEventArgs e)

{

DateTime kayitbas = newDateTime(2013, 8, 15);

DateTime kayitson = newDateTime(2013, 8, 31);

if (datePicker1.SelectedDate >= kayitbas &&
datePicker1.SelectedDate <= kayitson)

{

MessageBox.Show("Seçtiğiniz tarihte kayıt yapılabilir!");

}

else

{

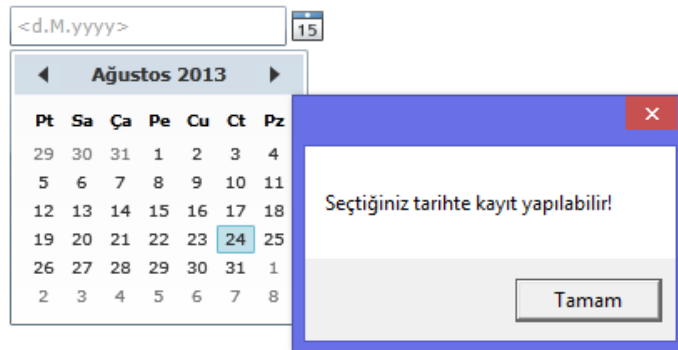
MessageBox.Show("Seçtiğiniz tarihte kayıt yapılamaz!");

}

}
```

➤ Çalışma anı görüntüsü

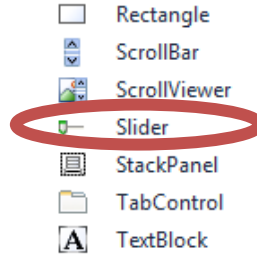
TARİH SEÇİNİZ:



Resim 1.14: Uygulama ekran görüntüsü

1.4.Sürgü(Slider)

Sürgü(Slider) çubuğunun sürgüsü kaydırıldığında bu kontrole ait “Value” özelliğinin değeri değişir ya da çubuğun sürgüsü “Value” özelliğine atanan değere göre hareket ettirilebilir.“Value” özelliğinin değerini programlama dili kodlarıyla kontrol ederek“Slider” istediğimiz bir amaca göre kullanabiliriz.



Resim 1.15: ToolBox üzerinde “Slider” kontrolü

“Slider” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
Value	“Slider” sürgüsü hareket ettirildiğinde bu özelliğin değeri değişir. Ayrıca “Value” özelliğine atanan değere göre de sürgü pozisyonu değiştirilir.	“Double” türünde sayısal veri tutar.
IsDirectionReversed	Sürgünün ilk konumunun çubuğun başında yada sonunda olmasını sağlar.	“Bool” türünde veri tutar.(True/False)
Minimum	“Value” özelliğinin alabileceği en küçük değeri belirler.	“Double” türünde sayısal veri tutar.
Maximum	“Value” özelliğinin alabileceği en büyük değeri belirler.	“Double” türünde sayısal veri tutar.
Orientation	“Slider” çubuğunu yatay veya dikey yönlendirebilmeyi sağlar.	Yatay için “Horizontal”, dikey için “Vertical” değeri atanmalıdır.
SmallChange	Klavye yön tuşları kullanılarak “Slider” sürgüsü kaydırıldığında “Value” özelliğinde olacak değişim miktarını belirler.	“Double” türünde sayısal veri tutar.
LargeChange	Slider çubuğu tıklanarak sürgü kaydırıldığında“Value” özelliğinde olacak değişim miktarını belirler.	“Double” türünde sayısal veri tutar.

Tablo 1.3: “Slider” özellikleri

“Slider” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

Örnek:

```
<Grid x:Name="LayoutRoot" Background="White">
<Slider Name="slider1" Maximum="10" Minimum="0" Width="100" Height="30"
/>
</Grid>
```

➤ Çalışma anı görüntüsü



Resim 1.16: Sürgü aracı

“Slider” kontrolünün “**ValueChanged**” olayı, “Value” özelliği her değiştiği anda çalışır. Bu olayın aldığı “e” parametresi “NewValue” ve “OldValue” isminde iki özellik barındırır. “NewValue” en son değeri, “OldValue” ise bir önceki değeri tutar.

Örnek: “Slider” sürgüsü her kaydırıldığında “Value” özelliğinin değişen değerini “TextBox” a yazdıran bir örneği inceleyelim.

➤ Tasarım görüntüsü

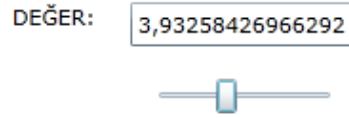


Resim 1.17: Metin kutusu ve sürgü araçları kullanımı

Program Kodları:“Slider”ın “ValueChanged” olayına yazılan kodlar:

```
privatevoid slider1_ValueChanged(object sender,  
RoutedPropertyChangedEventArgs<double> e)  
{  
  
    textBox1.Text = slider1.Value.ToString();  
  
}
```

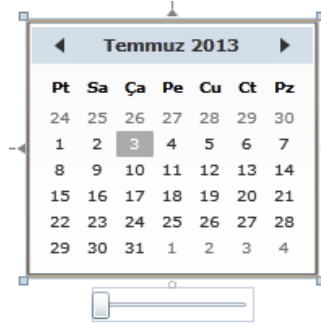
➤ Çalışma anı görüntüsü



Resim 1.18: Metin kutusu ve sürgü araçları kullanımı

Örnek: “Slider” ve “Calendar” içeren bir sayfada “Slider” sürgüsünü kaydirdığımızda takvimde görünen ayın değişmesini sağlayan bir örneği inceleyelim. (“Slider” için Maximum=12, Minimum=0 yapınız.)

➤ Tasarım görüntüsü



Resim 1.19: Takvim ve sürgü araçları kullanımı

Program Kodları: Kodlar içindeki açıklamalara dikkat ediniz.

```
public MainPage()
{
    InitializeComponent();

    slider2.Value = 1;

    //Value değerini 1 yaparak takvimde Ocak ayını görüntüler.
}

privatevoid slider2_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
{
    slider2.Value = Convert.ToInt32(slider2.Value);

    //Sürgünün oniki kademeli olması için Value özelliğinin
    //ondalık kısmı atılarak tekrar Value özelliğine atanıyor.
    int ay = Convert.ToInt32(slider2.Value);

    //Sürgü değeri ay değişkenine alınıyor
    string tarih = "1/" + ay + "/2013";

    //ay değişkeni tarih içine yerleştiriliyor.
    calendar1.DisplayDate = DateTime.Parse(tarih);

    //Takvimde tarih değişkeninin belirttiği tarihgörüntüleniyor.
}
```

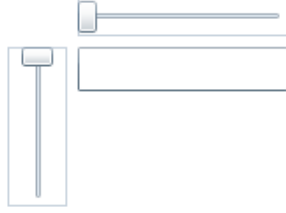
➤ Çalışma anı görüntüsü

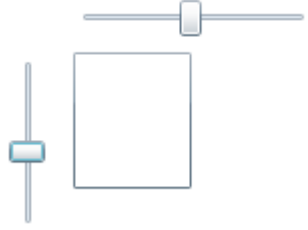


Resim 1.20: Takvim ve sürgü araçları kullanımı

UYGULAMA FAALİYETİ-1

Aşağıdaki işlem basamaklarını takip ederek “TextBox” kontrolünün genişlik ve yükseklik değerlerini iki adet “Slider” yardımıyla değiştiriniz.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir Silverlight projesi oluşturunuz.	➤ File→New→Project
➤ Sayfaya “TextBox” ekleyiniz.	➤ Width= “120” ➤ Height= “25” olarak ayarlayınız.
➤ Sayfaya “Slider” ekleyiniz.	➤ Maximum= “120” olarak ayarlayınız.
➤ Sayfaya ikinci bir “Slider“ ekleyiniz.	➤ Maximum= “120” ➤ IsDirectionReversed= “True” ➤ Orientation= “Vertical” olarak ayarlayınız.
➤ Kontrolleri hizalı hâle getiriniz.	
➤ “MainPage()”metodu içine “Slider” kontrollerinin başlangıç değerlerini atayan kodları yazınız.	<pre>public MainPage() { InitializeComponent(); slider1.Value = 120; slider2.Value = 25; }</pre>

➤ Her iki “Slider”ın“ValueChanged” olayları içine “TextBox”un genişlik ve yüksekliğini değiştiren kodları yazınız.	<pre>privatevoid slider1_ValueChanged(...) { textBox1.Width = slider1.Value; } privatevoid slider2_ValueChanged(...) { textBox1.Height = slider2.Value; }</pre>
➤ Projeyi çalıştırınız.	➤ F5 tuşunu kullanınız.
➤ “Slider” kontrollerinin sürgülerini kaydırarak “TextBox” boyutlarındaki değişimi gözlemleyiniz.	

UYGULAMA FAALİYETİ-2

Aşağıdaki işlem basamaklarını takip ederekiki “DatePicker” yardımıyla belirlenen bir tarih aralığının, “Calendar” üzerinde seçilemez duruma getirilmesi işlemini yapınız.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir Silverlight projesi oluşturunuz.	File→New→Project
➤ Sayfaya “Calendar” ekleyiniz.	ToolBox yada XAML kodlarını kullanabilirsiniz.
➤ Sayfaya iki adet “DatePicker” ekleyiniz.	ToolBox yada XAML kodları kullanabilirsiniz.
➤ Kontrolleri hizalı hâle getiriniz.	
➤ İkinci “DatePicker”in “SelectedDateChanged” olayına her iki “DatePicker”den seçilen tarih aralığını “Calendar” üzerinde seçilemez hâle getiren program kodlarını yazınız.	<pre>private void datePicker2_SelectedDateChanged(...) { CalendarDateRange aralık = new CalendarDateRange(datePicker1.SelectedDate .Value, datePicker2.SelectedDate.Value); calendar1.BlackoutDates.Add(aralık); }</pre>
➤ Projeyi çalıştırınız.	F5 tuşunu kullanınız.
➤ Her iki “DatePicker” üzerinde tarih seçimi yaparak “Calendar” üzerinde oluşan sonucu gözlemleyiniz.	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Metin kutusunu(TextBox) kullandınız mı?		
2. Takvimi(Calendar) kullandınız mı?		
3. Tarih seçiciyi(DatePicker) kullandınız mı?		
4. Sürgü(Slider)kontrolünü kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme’ ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Metin kutusu(TextBox) içeriğindeki metnin yazı tipini değiştiren özellik aşağıdakilerden hangisidir?
A) FontFamily
B) FontWeight
C) FontStyle
D) FontSize
2. Takvim(Calendar) üzerinde belirli bir tarihi seçmek için aşağıdaki özelliklerden hangisi kullanılmalıdır?
A) DisplayDate
B) SelectedDate
C) DisplayMode
D) SelectionMode
3. Tarih seçici(DatePicker) takvimi üzerinde seçili bir tarih değiştirildiği anda çalışan olay aşağıdakilerden hangisidir?
A) CalendarOpened
B) CalendarClosed
C) SelectedDateChanged
D) ValueChanged
4. Tarih seçici (DatePicker) takviminin, başlangıçta açık olarak görünüp görünmeyeceğini belirleyen özelliği aşağıdakilerden hangisidir?
A) BlackoutDates
B) FirstDayOfWeek
C) IsTodayHighLighted
D) IsDropDownOpen

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

5. Sürgü(Slider) çubuğunun sürgüsü kaydırıldığında bu kontrole aitözellikinin değeri değişir.
6.özellik sürgünün ilk konumunun, çubuğun başında ya da sonunda olmasını sağlar.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

İçerik kontrollerini kullanabileceksiniz.

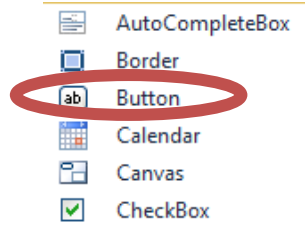
ARAŞTIRMA

- Düğme(Button),onay Kutusu(CheckBox),radyo düğmesi(RadioButton)“Popup” ve “ScrollView” kontrollerini araştırınız.

2. İÇERİK KONTROLLERİ

2.1.Düğme(Button)

Düğme(Button), üzerinde fareyle tıklandığında bir takım eylemler gerçekleştiren kontroldür.



Resim 2.1: ToolBox üzerinde Button kontrolü

“Button” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
Content	“Button” metnini değiştirmek için kullanılır.	“String” tipte veri tutar.
IsEnabled	“Button” kontrolünü aktif ya da pasif yapmak için kullanılır.	Aktif olması için “True”, pasif olması için “False” değerini almalıdır.
Width	Genişliği belirler.	“Double” türünde sayısal değer alır.
Height	Yüksekliği belirler.	“Double” türünde sayısal değer alır.
Foreground	Yazı rengini değiştirir.	

Tablo 2.1: “Button” özellikleri

Örnek: “Button” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

```
<Grid x:Name="LayoutRoot" Background="White">  
<ButtonContent="TIKLA" Height="40" Name="button1" Width="100" />  
</Grid>
```

➤ **Çalışma anı görüntüsü**



Resim 2.2: Düğme aracı

“Button” kontrolü için en önemli olay “Click” olayıdır. Düğmeye her tıklandığında “Click” olayının temsil ettiği metod çalışır.

Örnek: “Calendar”, “DatePicker” ve “Button” bulunan bir sayfada düğmeye tıklandığında, “DatePicker” üzerinden belirlediğimiz bir tarihi, takvim üzerinde seçilemez olarak işaretleyen bir uygulama örneğini inceleyelim.

➤ Tasarım görüntüsü



Resim 2.3: Tasarım görüntüsü

➤ Program kodları: "Button"un "Click" olayına yazılan kodlar:

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    CalendarDateRange tarih = new
    CalendarDateRange(datePicker1.SelectedDate.Value);

    calendar1.BlackoutDates.Add(tarih);

    MessageBox.Show("Seçtiğiniz tarih iptal edilmiştir!");
}
```

➤ Çalışma anı görüntüsü



Resim 2.4: Uygulama görüntüsü

“Button” özellikleri, programlama dili kodlarıyla dinamik olarak da değiştirilebilir.

Örnek: Sayfa açılışında düğmenin bazı özelliklerini değiştiren bir örneği inceleyelim.

```
public MainPage()
{
    InitializeComponent();

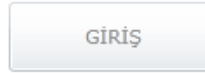
    button1.Width = 120;

    button1.Height = 40;

    button1.Content = "GİRİŞ";

    button1.IsEnabled = false;
}
```

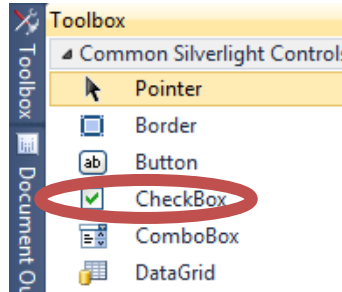
➤ Çalışma anı görüntüsü



Resim 2.5: Pasif buton görüntüsü

2.2. Onay Kutusu(CheckBox)

İşaretlenebilen bir kutucuktur. Onay kutusunun(CheckBox) işaret durumu program kodlarıyla kontrol edilerek farklı amaçlar için kullanılabilir. Sayfada bulunan onay kutularından biri, birkaçı ya da tamamı işaretlenebilir. Araç kutusu(Toolbox)ya da XAML kodlarını kullanarak sayfa üzerine ekleyip kullanabiliriz.



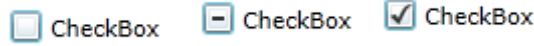
Resim 2.6: ToolBox üzerinde CheckBox kontrolü

“CheckBox” a ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
IsChecked	“CheckBox” işaretli olması durumunda “True”, işaretli olmadığı durumda “False” değerini alan özelliktir.	“Bool” türde veri tutar.
Content	Üzerindeki yazıyı değiştirir.	“String” tipte veri tutar.
IsEnabled	“CheckBox”u aktif ya da pasif yapmak için kullanılır.	Aktif olması için “True”, pasif olması için ise “False” değerini almalıdır.
IsThreeState	“CheckBox” kontrolünü üç durumlu ya da iki durumlu hâle getirir.	Üç durumlu olması için “True”, iki durumlu olması için ise “False” değerini almalıdır.

Tablo 2.2: “CheckBox” özellikleri

Not: CheckBox”un “IsThreeState” özelliği “True” yapıldığında üç farklı işaret durumuna sahip olacaktır. Aşağıda her üç durumda “CheckBox”un işareti ve “IsChecked” özelliğinin aldığı değerler gösterilmiştir.



IsChecked=false IsChecked=null

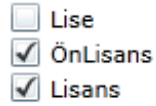
IsChecked=true

“CheckBox” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

Örnek:

```
<Grid x:Name="LayoutRoot" Background="White">
<StackPanel>
<CheckBox Content="Lise" IsChecked="False"/>
<CheckBox Content="ÖnLisans" IsChecked="True"/>
<CheckBox Content="Lisans" IsChecked="True"/>
</StackPanel>
</Grid>
```

➤ **Çalışma anı görüntüsü**



Resim 2.7: Onay Kutucuğu

“CheckBox” kontrolüne ait en önemli olaylar ise şunlardır:

- **Clickolayı:** “CheckBox” üzerinde her tıklandığında çalışır.
- **Checkedolayı:** “CheckBox” işaretlendiği anda çalışır.
- **UnCheckedolayı:** “CheckBox” işaretsiz hâle getirildiğinde çalışır.

Örnek: Bilgi girişi yapılan bir form sayfasında, “TextBox” kontrollerinden ikisinin, karşılarında bulunan birer “CheckBox” yardımıyla aktif yada pasif yapılabildiği bir örneği inceleyelim.

➤ Tasarım Görüntüsü

İSİM:

SOYİSİM:

ADRES: ☐ Girmek istemiyorum

TELEFON: ☐ Girmek istemiyorum

Resim 2.8:Tasarım görüntüsü

➤ Program kodları

1. **Yol:** Onay kutularının“Checked” ve “UnChecked” olayları kullanılmıştır.

```
privatevoid checkBox1_Checked(object sender, RoutedEventArgs e)
{
    textBox3.IsEnabled = false;
}
privatevoid checkBox1_Unchecked(object sender, RoutedEventArgs e)
{
    textBox3.IsEnabled = true;
}
```

2. Yol: Onay kutularının “Click” olayları kullanılmıştır.

```
privatevoid checkBox1_Click(object sender, RoutedEventArgs e)
{
    textBox3.IsEnabled = !(checkBox1.IsChecked.Value);
}

privatevoid checkBox2_Click(object sender, RoutedEventArgs e)
{
    textBox4.IsEnabled = !(checkBox2.IsChecked.Value);
}
```

Burada yazılan kodlara dikkat edelim. “textBox3” ve “textBox4” kontrolleri, karşısındaki “checkBox1” ve “checkBox2” kontrolleri işaretlendiğinde pasif olacaktır. Onun için !(mantıksal tersleme) operatörüyle her iki “ChecBox” kontrolünün işaret durumunun tersi her iki “TextBox” kontrolünün “IsEnabled” özelliğine atanmıştır. !operatörü kodların kısa olmasını sağlamıştır. Eğer bu operatörü kullanmasaydık “if” yapıları oluşturmamız gerekirdi.

➤ Çalışma anı görüntüsü



İSİM: Aziz

SOY İSİM: AKÇAY

ADRES: ☒ Girmek istemiyorum.

TELEFON: ☒ Girmek istemiyorum.

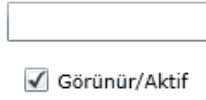
GÖNDER

Resim 2.9: Çalışma anı görüntüsü

Örnek:“CheckBox” aracının üç durumunu da kullanacağımız bir örnek yapalım. Bir “TextBox” ve bir “CheckBox” kullanacağız. “CheckBox”, ☒ işaretli olduğunda “TextBox” görünür ve aktif, ☐ işaretli olduğunda “TextBox” görünür ve pasif, işaretsiz (☐) olduğunda “TextBox” görünmez olacak ve üç duruma göre “CheckBox” üzerindeki yazı da değişecektir.

➤ **Tasarım görüntüsü**

Tasarım editöründe “CheckBox” için IsThreeState= “True”, IsChecked= “True”, ve Content=“Görünür/Aktif” yapalım.

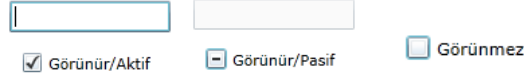


Resim 2.10: Tasarım görüntüsü

➤ Program kodları

```
privatevoid checkBox1_Click_1(object sender, RoutedEventArgs e)
{
    if (checkBox1.IsChecked == true)
    {
        textBox1.IsEnabled = true;           //Aktif
        textBox1.Visibility = Visibility.Visible; //Görünür
        checkBox1.Content = "Görünür/Aktif";
    }
    elseif (checkBox1.IsChecked == null)
    {
        textBox1.IsEnabled = false;          //Pasif
        textBox1.Visibility = Visibility.Visible; //Görünür
        checkBox1.Content = "Görünür/Pasif";
    }
    elseif (checkBox1.IsChecked == false)
    {
        textBox1.Visibility = Visibility.Collapsed; //Gizli
        checkBox1.Content = "Görünmez";
    }
}
```

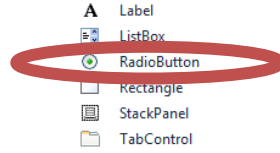
➤ Çalışma anı görüntüsü



Resim 2.11: Çalışma anı görüntüleri

2.3. Radyo Düğmesi(RadioButton)

Radyo düğmesi de(RadioButton) “CheckBox”gibi işaretlenebilen bir düğmedir. “CheckBox” tan farklı olarak sayfada bulunan radyo düğmelerinden sadece biri işaretlenebilir. Fakat aynı amaç için kullandığımız radyo düğmelerini bir grup ismi altında toplayabiliriz. Bu sayede birbirinden bağımsız radyo düğmesi grupları oluşturulur. Bu durumda her gruptan sadece biri işaretlenebilir. “RadioButton” un işaretlenme durumu da program kodlarıyla kontrol edilerek farklı amaçlar için kullanılabilir.



Resim 2.12: ToolBox üzerinde RadioButton kontrolü

“RadioButton” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
IsChecked	“RadioButton” işaretli olması durumunda “True”, işaretli olmaması durumunda “False” değerini alan özelliktir.	“Bool” türde veri tutar.
Content	“RadioButton”metnini değiştirmek için kullanılır.	“String” tipte veri tutar.
GroupName	Radyo düğmelerini bir grup hâline getirmek için grupta yer alacak her bir radyo düğmesinin “GroupName” özelliğine aynı grup ismi atanır.	“String” tipte veri tutar.
IsEnabled	Aktif ya da pasif yapmak için kullanılır.	Aktif olması için “True”, pasif olması için “False” değerini almalıdır.
IsThreeState	“RadioButton”u üç durumlu ya da iki durumlu hâle getirir.	Üç durumlu olması için “True”, iki durumlu olması için ise “False” değerini almalıdır.

Tablo 2.3: “RadioButton” özellikleri

Örnek: XAML kodları ile iki “RadioButton” grubu oluşturalım.

```
<Grid x:Name="LayoutRoot" Background="White">
    <StackPanel>
        <TextBlock Text="ASKERLİK DURUMU:"/>
        <RadioButton Content="YAPTI" GroupName="askerlikdugme"/>
        <RadioButton Content="YAPMADI" GroupName="askerlikdugme"/>
        <RadioButton Content="MUAF" GroupName="askerlikdugme"/>
        <TextBlock Text="MEZUNİYET DERECEŚİ:"/>
        <RadioButton Content="LİSE" GroupName="mezuniyetdugme"/>
        <RadioButton Content="ÖN LİSANS" GroupName="mezuniyetdugme"/>
        <RadioButton Content="LİSANS" GroupName="mezuniyetdugme"/>
    </StackPanel>
</Grid>
```

➤ **Çalışma anı görüntüsü**

ASKERLİK DURUMU:
☒ YAPTI
☐ YAPMADI
☐ MUAF
MEZUNİYET DERECEŚİ:
☒ LİSE
☐ ÖN LİSANS
☐ LİSANS

Resim 2.13: Radyo buton kullanımı

“RadioButton” aracına ait en önemli olaylar şunlardır:

- **Clickolayı:** “RadioButton” üzerinde her tıklandığında çalışır.
- **Checkedolayı:** “RadioButton” işaretlendiği anda çalışır.
- **Uncheckedolayı:** “RadioButton” işaretsiz hâle geldiği anda çalışır.

Örnek: İki “RadioButton” tarafından bir “TextBox”un aktif/pasif yapılabildiği bir örneği inceleyelim.

➤ **Tasarım görüntüsü**



Resim 2.14: Radyo buton ve metin kutusu kullanımı

➤ **Program kodları**

```
public MainPage()
{
    InitializeComponent();

    radioButton1.IsChecked = true; //Başlangıçta işaretleniyor.
}

privatevoid radioButton1_Checked(object sender, RoutedEventArgs e)
{
    textBox1.IsEnabled = true;
}

privatevoid radioButton2_Checked(object sender, RoutedEventArgs e)
{
    textBox1.IsEnabled = false;
}
}
```

➤ **Çalışma anı görüntüsü**



Resim 2.15: Çalışma anı görüntüsü

2.4. Popup

Açılır pencereler oluşturmak için kullanılır. “Popup”, kendi içine dâhil edilen kontrolleri bir açılır pencerede gösterir. “Popup” pencerelerinin açılıp kapanması program kodlarıyla yönetilir. Programlama dili kodlarıyla, sayfadaki çeşitli eylemlere bağlı olarak açılıp kapanması sağlanır. “ToolBox” üzerinden erişemediğimiz “Popup” kontrolü, XAML kodları kullanılarak tasarlanır.

“Popup” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
Name	“Popup” kontrolünün adını temsil eder. “Popup” kontrolünün mutlaka bir adı olmalıdır. Çünkü “Popup” penceresinin açılıp kapanması program kodları ile yönetilir. Bunun için programlama dili içinden bu kontrole erişebiliyor olmamız lazımdır.	“String” türde veri tutar.
IsOpen	“Popup” penceresinin gösterilmesini ve gizlenmesini sağlar. Daha çok programlama dili içinden dinamik olarak değer atadığımız IsOpen özelliği, “true” değeri aldığında “popup” penceresini açılır, “false” değeri aldığında ise “popup” penceresi kapanır.	“Bool” türde veri tutar.

Tablo 2.4: “Popup” özellikleri

Örnek: Fare işaretçisi “TextBox” üzerine getirildiğinde açılır pencere görüntüleyen bir örneği inceleyelim.

```
<StackPanel Margin="20">

<TextBox Height="23" Name="textBox1" Width="120"
MouseEnter="textBox1_MouseEnter" MouseLeave="textBox1_MouseLeave" />

<Popup x:Name="popup1">

<Border BorderThickness="2" CornerRadius="10" BorderBrush="Black">

<TextBlock Margin="10" Text="Adınızı buraya giriniz!"/>

</Border>

</Popup>

</StackPanel>
```

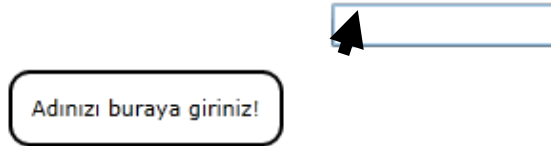
Sarı ile vurgulanmış olan XAML kodları, “Adınızı buraya giriniz!” metnini çerçeve içine alarak bu kontrolleri “Popup” içine dâhil eder. İsim olarak da “popup1” verilmiştir.

“TextBox” un XAML kodlarına da dikkat edelim. “Popup” penceresi fare “TextBox” üzerine geldiğinde açılacak, fare ayrıldığında ise kapanacaktır. Bunun için “TextBox” un “MouseEnter” ve “MouseLeave” olaylarına metot ismi atanmıştır. Aşağıda ise bu metotlara yazılan ve “Popup” penceresini açıp kapatan program kodları gösterilmiştir.

```
privatevoid textBox1_MouseEnter(object sender, MouseEventArgs e)
{
    popup1.IsOpen = true; //Popup penceresini açar.
}

privatevoid textBox1_MouseLeave(object sender, MouseEventArgs e)
{
    popup1.IsOpen = false; //Popup penceresini kapatır.
}
```

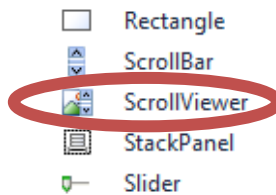
➤ Çalışma anı görüntüsü



Resim 2.16: Fare işaretçisinin “TextBox” üzerinde olduğu durum

2.5. ScrollViewer

İçine yerleştirilen nesnelerin boyutlarına göre yatay ve dikey kaydırma çubukları oluşturabilen bir yerleşim(layout) kontrolüdür.



Resim 2.17: ToolBox üzerinde ScrollViewer kontrolü

“ScrollView” kontrolüne ait sık kullanılan özellikler şunlardır.

Özellik Adı	Görevi	Açıklama
VerticalScrollBarVisibility	Dikey kaydırma çubuğunun görünürlüğünü ayarlar.	Tablonun altındaki notu inceleyiniz.
HorizontalScrollBarVisibility	Yatay kaydırma çubuğunun görünürlüğünü ayarlar.	

Tablo 2.5: “ScrollView” özellikleri

Not: “VerticalScrollBarVisibility” ve “HorizontalScrollBarVisibility” özelliklerinin alabileceği değerler şunlardır:

Disabled: Kaydırma çubuğu pasif yapar.

Auto: Kaydırma çubuğunun gerektiğinde otomatik olarak görüntülenmesini sağlar.

Visible: Kaydırma çubuğunun devamlı olarak görüntülenmesini sağlar.

Hidden: Kaydırma çubuğunun devamlı olarak gizlenmesini sağlar.

Örnek: Resim içeren bir “ScrollView” örneğini inceleyelim.

```
<Canvas>

<ScrollView Name="scrollView1" Width="150" Height="150"
HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto">

<Image Source="cicek.jpg" Width="400" Height="400"/>

</ScrollView>

</Canvas>
```

➤ Çalışma anı görüntüsü



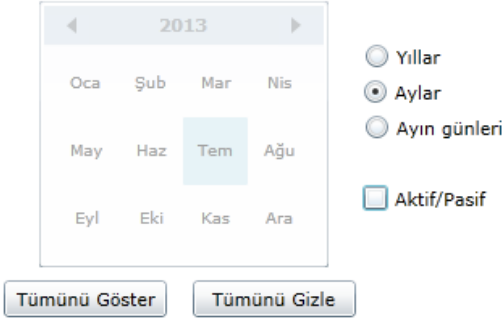
Resim 2.18: Kaydırma çubuklarının kullanımı

UYGULAMA FAALİYETİ

Bu uygulama faaliyetinde, üç radyo düğmesi kullanılarak bir takvimin görüntü modları değiştirilecektir. Bir onay kutusu kullanarak takvimin aktif/pasif olma durumu değiştirilecektir. Ayrıca iki düğme bulunacak, bir düğme tıklandığında düğmeler dışında tüm kontroller gizlenecek, diğeri tıklandığında tüm kontroller gösterilecektir. Açıklamaları dikkate alarak uygulama faaliyetini gerçekleştiriniz.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir proje oluşturunuz.	➤ File→New→Project
➤ Sayfaya “Calendar” ekleyiniz.	➤ ToolBox kullanınız.
➤ Sayfaya üç “RadioButton” ekleyip “Content” özelliklerini değiştiriniz.	☐ Yıllar ☐ Aylar ☐ Ayın günleri
➤ Sayfaya bir “CheckBox” ekleyip “Content” özelliğini belirleyiniz.	☐ Aktif/Pasif
➤ Sayfaya iki “Button” ekleyip “Content” özelliklerini belirleyiniz.	Tümünü Gizle Tümünü Göster
➤ Kontrolleri hizalı hâle getiriniz.	
➤ “MainPage()” metodunda “RadioButton” ve “CheckBox” için başlangıç işaretlerini atayan kodları yazınız.	<pre> public MainPage() { InitializeComponent(); radioButton3.IsChecked = true; checkBox1.IsChecked = true; } </pre>

➤ “RadioButton” ların“Checked” olaylarına takvimin görüntü modlarını değiştiren kodları yazınız.	<pre>privatevoid radioButton1_Checked(...) { calendar1.DisplayMode = CalendarMode.Decade; } privatevoid radioButton2_Checked(...) { calendar1.DisplayMode = CalendarMode.Year; } privatevoid radioButton3_Checked(...) { calendar1.DisplayMode = CalendarMode.Month; }</pre>
➤ “CheckBox” un “Click” olayına takvimin aktifliğini değiştiren kodları yazınız.	<pre>privatevoid checkBox1_Click(...) { calendar1.IsEnabled = checkBox1.IsChecked.Value; }</pre>
➤ Birinci “Button” un “Click” olayına nesneleri görünür yapan kodları yazınız.	<pre>privatevoid button1_Click(...) { calendar1.Visibility = System.Windows.Visibility.Visible; radioButton1.Visibility = System.Windows.Visibility.Visible; radioButton2.Visibility = System.Windows.Visibility.Visible; radioButton3.Visibility = System.Windows.Visibility.Visible; checkBox1.Visibility = System.Windows.Visibility.Visible; }</pre>

➤ İkinci “Button” un “Click” olayına nesneleri görünmez yapan kodları yazınız.	<pre>privatevoid button2_Click(...) { calendar1.Visibility = System.Windows.Visibility.Collapsed; radioButton1.Visibility = System.Windows.Visibility.Collapsed; radioButton2.Visibility = System.Windows.Visibility.Collapsed; radioButton3.Visibility = System.Windows.Visibility.Collapsed; checkBox1.Visibility = System.Windows.Visibility.Collapsed; }</pre>
➤ Projeyi çalıştırınız.	➤ F5 tuşunu kullanınız.
➤ Düğmeleri tıklayarak değişimleri gözlemleyiniz.	

UYGULAMA FAALİYETİ-2

Bu uygulama faaliyetinde bir “ScrollViewer” içinde bir resim(Image) bulunacaktır. Fare işaretçisi resim üzerine getirildiğinde resim dosyasının ismi, bir açılır pencerede(popup) görüntülenecektir.

Açıklamaları dikkate alarak uygulama faaliyetini gerçekleştiriniz.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir Silverlight projesi oluşturunuz.	➤ File→New→Project
➤ Sayfa üzerine “ScrollViewer” ekleyiniz.	➤ VerticalScrollBarVisibility = “Visible” ➤ HorizontalScrollBarVisibility = “Visible” yapınız.
➤ “ScrollViewer” içine “Image” ekleyiniz.	➤ Projenin “ClientBin” klasörüne bir resim kopyalayınız. ➤ “Image”nin “Source” özelliğine resim dosyasının adını atayınız. (Source=“cicek.jpg” gibi.)
➤ XAML tarafında“Popup” oluşturan kodları yazınız.	<pre> <Canvas Margin="20"> <ScrollViewer Name="scrollView1" Width="150" Height="150" HorizontalScrollBarVisibility="Visible" VerticalScrollBarVisibility="Visible"> <Image Source="cicek.jpg" Width="400" Height="400"/> </ScrollViewer> <Popup x:Name="popup1" Canvas.Top="150" Canvas.Left="150"> <Border BorderThickness="2" BorderBrush="Black"> <TextBlock Margin="10" Text="Dosya adı:cicek.jpg"/> </Border> </Popup> </Canvas> </pre>
➤ “Image”nin “MouseEnter” ve “MouseLeave” olaylarına “popup” penceresini açıp kapatan kodları yazınız.	<pre> privatevoid Image_MouseEnter(...) { popup1.IsOpen = true; } privatevoid Image_MouseLeave(...) { popup1.IsOpen = false; } </pre>

➤ Projeyi çalıştırınız.	➤ F5 tuşunu kullanınız.
➤ Fareyi resmin üzerine getirerek sonuçları gözlemleyiniz.	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Düğme(Button) kullandınız mı?		
2. Onay kutusu(CheckBox) kullandınız mı?		
3. Radyo düğmesi(RadioButton) kullandınız mı?		
4. Popup kullandınız mı?		
5. ScrollViewer kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Düğme(Button) metnini değiştiren özellik aşağıdakilerden hangisidir?
A) Content
B) Text
C) Caption
D) Value
2. Onay kutusunun(CheckBox) işaret durumunu temsil eden özelliği aşağıdakilerden hangisidir?
A) IsEnabled
B) IsChecked
C) Content
D) IsThreeState
3. Onay kutusunun(CheckBox) üç ya da iki durumlu olmasını sağlayan özelliği aşağıdakilerden hangisidir?
A) IsEnabled
B) IsChecked
C) Content
D) IsThreeState
4. “CheckBox” işaretlendiği anda çalışan olay aşağıdakilerden hangisidir?
A) Click
B) Unchecked
C) Checked
D) Changed
5. Radyo düğmelerini grup hâline getirmek için kullanılan özelliği aşağıdakilerden hangisidir?
A) GroupName
B) Name
C) Content
D) IsChecked

6. “RadioButton”işaretsiz duruma geldiği andaçalışan olay aşağıdakilerden hangisidir?

- A) Click
- B) Unchecked
- C) Checked
- D) Changed

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

7. “Popup” penceresini açıp kapatan özelliği.....dir.

8. “ScrollView” kontrolünün.....özellği yatay kaydırma çubuğunun görünürlüğünü ayarlar.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-3

AMAÇ

Gelişmiş kontrolleri kullanabileceksiniz.

ARAŞTIRMA

- “ItemsControl”, “DataGrid” ve “ListBox” kontrollerini araştırınız.

3. GELİŞMİŞ KONTROLLER

3.1.ItemsControl Kullanımı

“ItemsControl”, bir veri kaynağından aldığı kayıtları üzerinde listeleyen kontroldür. Veri kaynağından alınan her kayıt, bir liste elemanı olarak karşımıza çıkar. “ItemsControl”, alışveriş sitelerindeki ürün katalogları gibi oldukça kullanışlı listeler hazırlamamızı sağlar. Bu kontrolü oluşturmak için XAML kodlarını kullanmalıyız. Sayfa üzerinde “ItemsControl” oluşturulurken önce veri kaynağındaki kayıt yapısına uygun biçimde, birkaç kontrol bir araya getirilerek liste elemanı şablonu tasarlanır.Daha sonra tüm liste elemanlarının yerleşeceği panelin şablonu tasarlanır.

A MARKASI	B MARKASI	C MARKASI	D MARKASI
			
1740 TL	1920 TL	900 TL	1100 TL
Stok Durumu:	Stok Durumu:	Stok Durumu:	Stok Durumu:
☐	☐	☒	☒

Resim 3.1: ItemsControl kullanımı

Yukarıdaki resimde gösterilen“ItemsControl” örneğinde, bilgisayarlar için marka, ürün resmi, fiyat ve stok durumu bilgilerinden oluşan kayıt yapısına uygun şekilde liste elemanı şablonu tasarlanmıştır. Bu şablon, marka ve fiyat bilgilerini temsil edeniki “TextBlock”, ürün resmini temsil eden bir “Image” ve stok durumu bilgisini temsil eden bir “CheckBox” kontrollerinden oluşmaktadır. Daha sonra “ItemsControl”, kayıtların bulunduğu veri kaynağına bağlandığında otomatik olarak bütün kayıtlar liste elemanları olarak görüntülenmiştir.

“ItemsControl”un sahip olduđu bazı önemli özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
ItemsSource	Listelenecek kayıtları içeren veri kaynağını bu özellik ile ilişkilendirmemiz gerekir.	
Items	Liste elemanlarının tutulduğu dizidir.	
ItemTemplate	Liste elemanları içinşablon tasarımı yapmayı sağlayan özelliktir.	

Tablo 3.1: “ItemsControl” özellikleri

Örnek: Bilgisayarlara ait bazı bilgiler içeren bir kayıt listesini “ItemsControl” üzerinde görüntüleyelim. Kayıt listesini, tanımlayacağımız bir sınıftan üreteceğiz.

1.Adım: Bilgisayarlara ait resimleri projenin “ClientBin” klasörüne kopyalayacağız.

2.Adım: Bilgisayarlar için marka, fiyat, fotoğraf ve stok durumu özelliklerini içeren bir sınıf(Class) yapısı oluşturacağız.

```
public class notebook
{
    public string marka { get; set; }
    public string fiyat { get; set; }
```

3.Adım: “notebook” sınıfından nesne listesi üreten metodu hazırlayacağız.

```
public List<notebook> NotebookKayitGetir()
{
    List<notebook> dizustubilgisayarlar = newList<notebook>()
    {
        new notebook { marka="A MARKASI", fiyat="1740 TL", fotografadi="A.jpg",
            stok=false },
        new notebook { marka="B MARKASI ", fiyat="1920 TL", fotografadi="B.jpg",
            stok=false },
        new notebook { marka="C MARKASI ", fiyat="900 TL", fotografadi="C.jpg",
            stok=false }
```

Bu metot hakkında şunları belirtmemiz gerekiyor:

- Metot “notebook” sınıfından ürettiği nesneleri liste hâlinde geri döndüreceği için `public List<notebook> NoteBookKayitGetir()` başlık tanımlaması yapılmıştır.
- Geri döndürülecek olan `List<notebook> dizustubilgisayarlar = newList<notebook>()` şeklinde metot içinde (yerel olarak) tanımlanmıştır.
- Liste içinde “notebook” sınıfından dört nesne üretilip nesnelerin ilgili özelliklerine marka, fiyat, fotoğraf ve stok durumu verileri aktarılması için `new notebook { marka="A MARKASI", fiyat="1740 TL", fotografadi="A.jpg", stok=false }` şeklinde dört kod satırı kullanılmıştır.
- Return bildirisi, oluşturulan listeyi geri döndürmektedir.

4.Adım: Sayfanın XAML tarafında “ItemsControl” yapısını oluşturup liste elemanı ve liste paneli şablonlarını tasarlayacağız.

```
<Canvas x:Name="LayoutRoot" Background="White">

<Border BorderBrush="Red" BorderThickness="3" CornerRadius="8" Canvas.Left="36"
Canvas.Top="53">

<ItemsControl x:Name="bilgisayarliste" Width="Auto" Height="Auto">
    <ItemsControl.ItemTemplate>
        <DataTemplate>
            <Border BorderBrush="Red" BorderThickness="1">
                <StackPanel Orientation="Vertical">
                    <TextBlock Text="{Binding marka}" HorizontalAlignment="Center"/>
                    <Image Source="{Binding fotografadi}" Width="100" Height="100"/>
                    <TextBlock Text="{Binding fiyat}" HorizontalAlignment="Center"/>
                    <TextBlock Text="Stok Durumu:" HorizontalAlignment="Center"/>
                    <CheckBox IsChecked="{Binding stok}" HorizontalAlignment="Center"/>
                </StackPanel>
            </Border>
        </DataTemplate>
    </ItemsControl>
</Border>
</Canvas>
```

“ItemsControl”un XAML yapısına ilişkin şunları belirtmemiz gerekiyor.

- <ItemsControl></ItemsControl>yapısıyla oluşturulan“ItemsControl”asayfanın C# tarafından erişebilmek için “bilgisayarliste” ismi verilmiştir.
- “ItemsControl”, temelde iki şablonbarındırır. Bunlar,liste elemanı şablonu(<ItemsControl.ItemTemplate>) vebütün liste elemanlarının yerleşeceği panelinşablonudur.(<ItemsPanelTemplate>)
- <ItemsControl.ItemTemplate>içinde<DataTemplate>yapısı oluşturulmuş ve<DataTemplate> içinde,kayıt yapısındaki alanları temsil edenkontrol grubu oluşturulmuştur. Örneğimizde “marka” ve “fiyat” bilgileri için iki“TextBlock”, “ürün resmi” için “Image” ve “stokta olma durumu” için “CheckBox” kullanılmıştır.Ayrıca bu kontroller bir “StackPanel” içinde alt alta yerleştirilmiştir. “StackPanel”de bir çerçeve(Border) içine alınmıştır. Bu sayede “ItemsControl”dakiliste elemanlarıbirbirinden çerçevelerle ayrılmış olur.
- <TextBlockText="{Binding marka}"/>satırında “notebook” sınıf yapısı içindeki “marka” özelliği ile “TextBlock”un “Text” özelliği bağlanmış ve bu sayede “TextBlock”un,“marka” alanını temsil etmesi sağlanmıştır.“CheckBox” ve “Image” içinde benzer durum geçerlidir.
- Bütün liste elemanlarının yerleşeceği panel,<ItemsControl.ItemsPanel>yapısı ile oluşturulur.Panele ait şablon ise<ItemsPanelTemplate>bloğunda oluşturulmuştur.
- Liste elemanlarının yan yana listelenmesi için<ItemsPanelTemplate>şablon yapısı içinde yatay yönlendirilmiş bir “StackPanel”kullanılmıştır.
- Ayrıca oluşturulan“ItemsControl”dabir “Border”içine alınmıştır.

5.Adım: Sayfanın açılışı sırasında“NoteBookKayitGetir()” metodunun ürettiği listeyi “ItemsControl” un “ItemsSource” özelliğine aktararak“ItemsControl” un veri bağlantısını kuracağız.

```
public MainPage()
{
    InitializeComponent();

    bilgisayarliste.ItemsSource = NoteBookKayitGetir();
}
```

➤ Çalışma anı görüntüsü



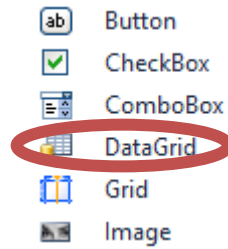
Resim3.2: Çalışma anı görüntüsü

3.2. DataGrid Kullanımı

Veri kaynaklarından alınan kayıtları listeleyebilen kontrollerden biri de “DataGrid” kontrolüdür. Satır ve sütun yapısına sahip olan “DataGrid”, kayıtları tablo hâlinde listeler. “Toolbox” ya da XAML kodları kullanarak sayfa üzerine ekleyip kullanabiliriz.

MARKA	MODEL	FİYAT	STOK
DELL Inspiron	DN800	1550 TL	☐
SAMSUNG	ATIV 3	1480 TL	☒
HP Compact	H780	1330 TL	☒
ACER	Aspire 5600	1200 TL	☐

Resim 3.3: Kayıt listeleyen bir DataGrid görüntüsü



Resim 3.4: ToolBox üzerinde “DataGrid” kontrolü

“DataGrid” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
ItemsSource	“DataGrid” e aktarılacak kayıt listesini bu özellik ile ilişkilendirmemiz gerekir.	
AutoGenerateColumns	Kayıt yapısındaki veri alanlarına göre “DataGrid” sütunlarının otomatik olarak oluşturulup oluşturulmayacağını belirler.	“Bool” tipte veri tutar.“True” değeri alırsa sütunlar otomatik olarak oluşturulur. “False” değeri alırsa sütunların kullanıcı tarafından oluşturulup veri bağlantılarının kurulması gerekir.
Columns	Sütun tanımlamalarının yapılabilmesi için alt özellikler içerir.	

Tablo 3.2: “DataGrid” özellikleri

“DataGrid” kontrolünün XAML yapısını bir örnek üzerinde inceleyelim.

Örnek:

```
<Grid x:Name="LayoutRoot" Background="White" Width="460">

<data:DataGrid AutoGenerateColumns="False" Height="Auto"
Name="dataGrid1" Width="Auto">

<data:DataGrid.Columns>

<data:DataGridTextColumn Header="MARKA" Width="Auto" Binding="{Binding
marka}" />

<data:DataGridTextColumn Header="FİYAT" Width="Auto" Binding="{Binding
fiyat}" />

<data:DataGridCheckBoxColumn Header="STOK"
Width="Auto" Binding="{Binding stok}" />

</data:DataGrid.Columns>

</data:DataGrid>

</Grid>
```

“DataGrid”in XAML yapısı ile ilgili olarak şunları belirtmemiz gerekiyor:

- <data:DataGrid></data:DataGrid>etiketleri içerisinde oluşturulan “DataGrid”in programlanabilmesi için “Name” özelliğine bir isim verilmelidir.
- “DataGrid” sütunları <data:DataGrid.Columns>yapısı içinde oluşturulur.
- “DataGrid”üzerinde üç tipte sütun oluşturulabilir.
 - **DataGridTextColumn:** Kayıtlardaki “marka”, “fiyat” gibi metin türündeki alanların içeriğini göstermek için kullanılır. <data:DataGridTextColumn />etiketiyle oluşturulur.
 - **DataGridCheckBoxColumn:** Kayıtlardaki stokta var olma durumu gibi “bool(true/false)” türdeki alanları temsil etmek için kullanılır. <data:DataGridCheckBoxColumn />etiketiyle oluşturulur.
 - **DataGridTemplateColumn:** Birkaç kontrolden oluşan özel sütun şablonu tasarlamak için kullanılır. Örneğin bir sütunda kişilere ait fotoğrafların üst kısmında ayrıca kimlik numaraları kullanılacaksa o sütunda hem “Image” hem de “TextBlock” kontrolleri beraber

bulunacaktır. Bu şekilde istediğimiz kontrol bileşimlerini kullanarak farklı sütun tasarımları yapmak için “DataGridTemplateColumn” etiketi kullanılır. `<data:DataGridTemplateColumn>` bloğu içinde oluşturulur.

- Oluşturulan sütunların “Binding” özelliği kullanılarak (Binding="{Binding marka}" gibi) kayıt yapısındaki ilgili alanlara bağlanması gerekir.
- Sütunların genişliğinin, içeriğine göre otomatik değişmesi için Width="Auto" yapılmıştır.
- **Program kodları**

```
public class notebook
{
    public string marka { get; set; }
    public string fiyat { get; set; }
    public bool stok { get; set; }
}

public List<notebook> NoteBookKayitGetir()
{
    List<notebook> dizustubilgisayarlar = newList<notebook>(){
        new notebook {marka="A MARKASI", fiyat="1740 TL", stok=true},
        new notebook {marka="B MARKASI ", fiyat="1920 TL", stok=false},
        new notebook {marka="C MARKASI ", fiyat="900 TL", stok=false},
        new notebook {marka="D MARKASI ", fiyat="1100 TL", stok=true}
    }
}
```

Buna benzer program kodları hakkında “ItemsControl Kullanımı” başlığı altında gerekli bilgiler verilmişti.

➤ Çalışma anı görüntüsü

MARKA	FİYAT	STOK
Samsung ATIV Book 5	1740 TL	☒
Asus N56VZ	1920 TL	☐
Casper CN TKP 847b	900 TL	☐
Acer E1531	1100 TL	☒

Resim 3.5: Çalışma anı görüntüsü

“DataGrid” sütunlarını tasarlamadan, “AutoGenerateColumns” özelliğini “True” yaparsak sütunlar, “DataGrid”e aktarılan kayıtların veri yapısındaki alanlara göre otomatik olarak oluşturulur. Bu pratik kullanımı bir örnek üzerinde gösterelim.

Örnek:

```
<data:DataGrid AutoGenerateColumns="True" Height="Auto" Name="dataGrid1"
Width="Auto" />
```

C# tarafındaki kodlar bir önceki örnekle aynıdır.

➤ Çalışma anı görüntüsü

marka	fiyat	stok
Samsung ATIV Book 5	1740 TL	☒
Asus N56VZ	1920 TL	☒
Casper CN TKP 847b	900 TL	☒
Acer E1531	1100 TL	☒

Resim 3.6: Çalışma anı görüntüsü

Örnek: “FİYAT” sütununda şablon(DataGridTemplateColumn) oluşturulan ve aşağıda görüntüsü verilen “DataGrid” örneğini inceleyelim.

MARKA	STOK	FİYAT
Samsung ATIV Book 5	☒	 1740 TL
Asus N56VZ	☒	 1920 TL
Casper CN TKP 847b	☒	 900 TL
Acer E1531	☒	 1100 TL

Resim 3.7: Çalışma anı görüntüsü

➤ **XAML kodları**

```
<Grid x:Name="LayoutRoot" Background="White" Width="460">
<data:DataGrid AutoGenerateColumns="False" Height="Auto" Name="dataGrid1"
Width="Auto">
<data:DataGrid.Columns>
<data:DataGridTextColumn Header="MARKA" Width="Auto" Binding="{Binding marka}"
/>
<data:DataGridCheckBoxColumn Header="STOK" Width="Auto" Binding="{Binding
stok}" />
<data:DataGridTemplateColumn Header="FİYAT">
<data:DataGridTemplateColumn.CellTemplate>
<DataTemplate>
<StackPanel>
```

XAML kodlarındaiç içe “DataGridTemplateColumn”, “CellTemplate” ve “DataTemplate” yapıları oluşturulmuştur. “DataTemplate” içinde ise “Image” ve “TextBlock” barındıran bir “StackPanel” oluşturulmuştur. “Image” ve “TextBlock” için “Binding” özellikleri kullanılarak veri bağlantıları yapılmıştır.

➤ Program kodları

```
public class notebook
{
    public string marka { get; set; }
    public string fiyat { get; set; }
    public string fotografadi { get; set; }
    public bool stok { get; set; }
}

public List<notebook> NoteBookKayitGetir()
{
    List<notebook> dizustubilgisayarlar = newList<notebook>()
    {
        new notebook { marka="Samsung ATIV Book 5", fiyat="1740 TL",
            fotografadi="samsung.jpg", stok=true},
        new notebook { marka="Asus N56VZ", fiyat="1920 TL", fotografadi="asus.jpg",
            stok=true},
        new notebook { marka="Casper CN TKP 847b", fiyat="900 TL",
            fotografadi="casper.jpg", stok=true},
        new notebook { marka="Acer E1531", fiyat="1100 TL", fotografadi="acer.jpg",
            stok=true}
    };
    return dizustubilgisayarlar;
}

public MainPage()
{
    InitializeComponent();
    dataGrid1.ItemsSource = NoteBookKayitGetir();
}
```

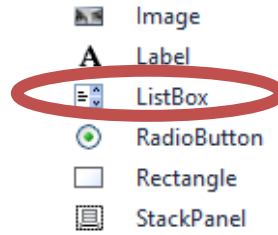
➤ Çalışma anı görüntüsü

MARKA	STOK	FIYAT
Samsung ATIV Book 5	☒	 1740 TL
Asus N56VZ	☒	 1920 TL
Casper CN TKP 847b	☒	 900 TL
Acer E1531	☒	 1100 TL

Resim 3.8: Çalışma anı görüntüsü

3.3. ListBox Kullanımı

Verileri sayfada listelemek için kullandığımız kontrollerden birisi de “ListBox” dur. Sadece metinleri listeleyerek basit şekilde kullanabileceğimiz gibi kayıt yapısına uygun şekilde şablon tasarımı yaparak daha işlevsel hâle de getirebiliriz. (“ItemsControl” da olduğu gibi)



Resim 3.9: “ToolBox” üzerinde “ListBox” kontrolü

“ListBox” kontrolüne ait sık kullanılan özellikler şunlardır:

Özellik Adı	Görevi	Açıklama
ItemsSource	Listelenecek kayıtları içeren veri kaynağını temsil eder.	
Items	Liste elemanlarının tutulduğu dizidir.	“ListBoxItem” türünde bir dizidir.
SelectedIndex	Liste elemanlarının ardışık index numaraları vardır. “SelectedIndex” özelliği ise seçili olan liste elemanının index numarasını tutar.	Liste kutusunda seçili durumda hiçbir liste elemanı olmadığında bu özellik -1 olur.
SelectedItem	Seçili durumda bulunan elemanı temsil eder.	
ItemTemplate	Liste elemanı için şablon tasarımı yapmak için kullanılır.	

Tablo 3.3: “ListBox” özellikleri

“ListBox” elemanları “ListBoxItem” sınıfı türündedir. “ListBoxItem” sınıfının önemli özellikleri ise şunlardır:

Özellik Adı	Görevi	Açıklama
Content	Liste elemanının içeriğini belirler.	
IsEnabled	Liste elemanını aktif veya pasif yapar. Aktif yapmak için “true”, pasif yapmak için “false” değerini alır.	“Bool” türde veri tutar.
IsSelected	Liste elemanının seçili olmadurumunu belirlemek için kullanılır. “true” ise seçilidir, “false” ise seçili değildir.	“Bool” türde veri tutar.

Tablo 3.4: “ListBoxItem” sınıfı özellikleri

Örnek: Liste elemanlarının bir “Image” ve iki “TextBlock” içerdiği bir “ListBox” yapısını inceleyelim.

➤ XAML yapısı

```
<Canvas x:Name="LayoutRoot" Background="White">

<ListBox Height="268" x:Name="listBox1" Width="324">

<ListBoxItem>

<StackPanel Orientation="Horizontal">

<Image Source="uygun.jpg" Width="60" Height="60" Margin="10" />

<TextBlock Text="ahmet09" VerticalAlignment="Center" Margin="10" />

<TextBlock Text="UYGUN" VerticalAlignment="Center" Margin="10" />

</StackPanel>

</ListBoxItem>

<ListBoxItem>

<StackPanel Orientation="Horizontal">

<Image Source="uygun.jpg" Width="60" Height="60" Margin="10" />

<TextBlock Text="veli-35" VerticalAlignment="Center" Margin="10" />

<TextBlock Text="UYGUN" VerticalAlignment="Center" Margin="10" />

</StackPanel>

</ListBoxItem>

<ListBoxItem>

<StackPanel Orientation="Horizontal">

<Image Source="uygundegil.jpg" Width="60" Height="60" Margin="10" />

<TextBlock Text="hakan-07" VerticalAlignment="Center" Margin="10" />

<TextBlock Text="UYGUN DEĞİL" VerticalAlignment="Center" Margin="10" />

</StackPanel>

</ListBoxItem>

</ListBox>
```

Bu kodlardan da anlaşılacağı gibi;

- “ListBox” kontrolü `<ListBox></ListBox>` etiketleri içinde oluşturulur.
- Liste elemanları `<ListBoxItem></ListBoxItem>` içinde tasarlanır.
- Liste elemanları istenilen sayıda kontrol barındırabilir.
- **Çalışma anı görüntüsü**



Resim 3.10: Çalışma anı görüntüsü

“ListBox” bu şekilde statik verileri listeleyebildiği gibi bir veri kaynağından alınan verileri de listeleyebilir (“ItemsControl” ve “DataGrid” gibi).

Örnek: Bir veri kaynağından aldığı kayıtları listeleyen bir “ListBox” örneğini inceleyelim.

➤ XAML yapısı

```
<Canvas x:Name="LayoutRoot" Background="White">

<ListBox Height="245" x:Name="listBox1" Width="291">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<Image Source="{Binding resim}" Width="50" Height="50" Margin="5"/>

<TextBlock Text="{Binding ulkeadi}" Margin="5" VerticalAlignment="Center"/>

<TextBlock Text="{Binding nufus}" Margin="5" VerticalAlignment="Center"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Canvas>
```

“ItemsControl” ile “ListBox” kontrollerinin şablon tasarımı hemen hemen aynıdır.<ItemTemplate>içindeki<DataTemplate>bloğuna kullanılması gerekli olan “StackPanel”, “Image”, “TextBlock” gibi kontroller eklenip veri bağlantıları yapılır.

➤ Program kodları

```
public class ulke
{
    public string ulkeadi { get; set; }
    public int nufus { get; set; }
    public string resim { get; set; }
}

public List<ulke> ulkelerigetir()
{
    List<ulke> ulkeler = newList<ulke>{
        new ulke { ulkeadi="Türkiye", nufus=81456451, resim="tur.png"},
        new ulke { ulkeadi="Pakistan", nufus=110873124, resim="pak.png"},
        new ulke { ulkeadi="Hindistan", nufus=1221431789, resim="ind.png"},
        new ulke { ulkeadi="Japonya", nufus=140342987, resim="jpn.png"}
    };
    return ulkeler;
}

public MainPage()
{
    InitializeComponent();
    listBox1.ItemsSource = ulkelerigetir();
}
```

Görüldüğü gibi “ulkelerigetir()” metodundan gelen liste, “ListBox” kontrolünün “ItemsSource” özelliğine aktarılmıştır.

➤ Çalışma anı görüntüsü



Resim 3.11: Çalışma anı görüntüsü

“ListBox”un“Items” sınıfı altında yer alan önemli birkaç metod ise şunlardır:

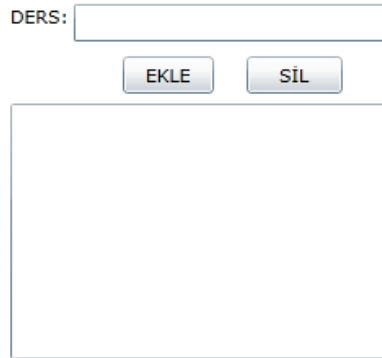
Metod Adı	Görevi	Açıklama
Add(Object)	“ListBox”a yeni bir eleman ekler.	
Remove(Object)	İsmi belirtilen elemanı siler.	
RemoveAt(index)	Index numarası belirtilen elemanı siler.	

Tablo 3.5: “ListBox” metodları

Not: ListBox” kontrolünün en önemli olayı “**SelectionChanged**” olayıdır. Bu olay bir eleman seçildiği anda çalışır.

Örnek: Liste kutusuna eleman ekleyip silen birörneği inceleyelim.

➤ Tasarım görüntüsü



Resim 3.12: Tasarım görüntüsü

➤ **Program kodları**

```
privatevoid button1_Click(object sender, RoutedEventArgs e)
{
    listBox1.Items.Add(textBox1.Text);
}

privatevoid button2_Click(object sender, RoutedEventArgs e)
{
    listBox1.Items.Remove(listBox1.SelectedItem);
}

privatevoid listBox1_SelectionChanged(object sender, SelectionChangedEventArgs
e)
{
    textBox1.Text = listBox1.SelectedItem.ToString();
}
```

- “TextBox”a girilen bilgiyi “ListBox”a eklemek için “EKLE” düğmesinin “Click” olayında: listBox1.Items.Add(textBox1.Text);kod satırı kullanılmıştır.
- “ListBox”ta seçili olan elemanı silmek için “SİL” düğmesinin “Click” olayında:listBox1.Items.Remove(listBox1.SelectedItem);kod satırı kullanılmıştır. Bilindiği gibi “ListBox” üzerinde seçili olan eleman “SelectedItem” özelliğinde tutulur.
- “ListBox”tan seçilen elemanın “TextBox”tagösterilmesi için“ListBox”un “SelectedItemChanged” olayında: textBox1.Text = listBox1.SelectedItem.ToString();kod satırı kullanılmıştır.

➤ **Çalışma anı görüntüsü**

DERS:

Matematik

Fizik

Kimya

Tarih

Dil Ve Anlatım

Mesleki Gelişim

Resim 3.13:Çalışma anı görüntüsü

UYGULAMA FAALİYETİ

Bu uygulama faaliyetinde bir grup öğrencinin bir dersten geçme/kalma durumlarını hem “ItemsControl”, hem “DataGrid” hem de “ListBox” üzerinde listeleyeceğiz. Açıklamaları dikkate alarak uygulama faaliyetini gerçekleştiriniz.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir proje oluşturunuz.	➤ File→New→Project
➤ Uygulamada kullanacağımız ikonları “ClientBin” klasörüne kopyalayınız.	  
➤ “ogrenci” isminde bir sınıf(Class) yapısı oluşturunuz.	<pre>public class ogrenci { public string isimsoyisim { set; get; } public int numara { set; get; } public string durum { set; get; } public string durumikon { set; get; } }</pre>
➤ XAML tarafında “Canvas” içinde bir “ItemsControl” yapısı oluşturunuz.	<pre><ItemsControl x:Name="ogrenciliste" Canvas.Left="133" Canvas.Top="105"> <ItemsControl.ItemTemplate> <DataTemplate> <StackPanel Orientation="Vertical" Margin="5"> <TextBlock Text="{Binding isimsoyisim}" HorizontalAlignment="Center"/> <Image Source="ogrenci.jpg" Width="50" Height="50"/> <TextBlock Text="{Binding numara}" HorizontalAlignment="Center"/> <TextBlock Text="{Binding durum}" HorizontalAlignment="Center"/> <Image Source="{Binding durumikon}" Width="25" Height="25"/> </StackPanel> </DataTemplate> </ItemsControl.ItemTemplate> </ItemsControl></pre>

	<pre> </StackPanel> </DataTemplate> </ItemsControl.ItemTemplate> <ItemsControl.ItemsPanel> <ItemsPanelTemplate> <StackPanel Orientation="Horizontal"/> </ItemsPanelTemplate> </ItemsControl.ItemsPanel> </ItemsControl> </pre>
➤ Veri kaynağını oluşturan ve “ItemsControl” a bağlayan program kodlarını yazınız.	<pre> public List<ogrenci> ogrencilisteal() { List<ogrenci> ogrenciler = newList<ogrenci>{ newogrenci { isimsoyisim="İbrahim GÜNDOĞAN", numara=102, durum="GEÇTİ", durumikon="gecti.jpg"}, newogrenci { isimsoyisim="Hasan AKÇIL", numara=105, durum="KALDI", durumikon="kaldi.jpg"}, newogrenci { isimsoyisim="Musa ÖZBAY", numara=107, durum="GEÇTİ", durumikon="gecti.jpg"} }; return ogrenciler; } public MainPage() { InitializeComponent(); ogrenciliste.ItemsSource = ogrencilisteal(); } </pre>

	}
➤ Uygulamayı çalıştırınız(F5). Sonuçları gözlemleyiniz.	İbrahim GÜNDOĞAN Hasan AKÇİL Musa ÖZBAY  102 GEÇTİ 105 KALDI 107 GEÇTİ 
➤ “ItemsControl”u silip“DataGrid” ekleyin ve “DataGrid” için gerekli XAML yapısını oluşturun.	<pre> <data:DataGrid AutoGenerateColumns="False" Canvas.Left="91" Canvas.Top="44" Height="Auto" Name="dataGrid1" Width="Auto" > <data:DataGrid.Columns> <data:DataGridTextColumn Header="İSİM/SOYİSİM" Width="Auto" Binding="{Binding isimsoyisim}" /> <data:DataGridTextColumn Header="DURUM" Width="Auto" Binding="{Binding durum}" /> <data:DataGridTemplateColumn> <data:DataGridTemplateColumn.CellTemplate> <DataTemplate> <Image Source="{Binding durumikon}" Width="25" Height="25"/> </DataTemplate> </data:DataGridTemplateColumn.CellTemplate> </data:DataGridTemplateColumn> </data:DataGrid.Columns> </data:DataGrid> </pre>
➤ Veri listesini “DataGrid”üzerinde listelemek için “MainPage()”metodundager ekli kodları yazınız.	<pre> public MainPage() { InitializeComponent(); dataGrid1.ItemsSource = ogrencilisteal(); } </pre>

➤ Uygulamayı çalıştırınız(F5). Sonuçları gözlemleyiniz.	<table><tr><th>İSİM/SOYİSİM</th><th>DURUM</th><th></th></tr><tr><td>İbrahim GÜNDOĞAN</td><td>GEÇTİ</td><td>✓</td></tr><tr><td>Hasan AKÇİL</td><td>KALDI</td><td>✗</td></tr><tr><td>Musa ÖZBAY</td><td>GEÇTİ</td><td>✓</td></tr></table>	İSİM/SOYİSİM	DURUM		İbrahim GÜNDOĞAN	GEÇTİ	✓	Hasan AKÇİL	KALDI	✗	Musa ÖZBAY	GEÇTİ	✓
İSİM/SOYİSİM	DURUM												
İbrahim GÜNDOĞAN	GEÇTİ	✓											
Hasan AKÇİL	KALDI	✗											
Musa ÖZBAY	GEÇTİ	✓											
➤ “DataGrid”i silip bir “ListBox” ekleyinve “ListBox”için gerekli XAML yapısını oluşturun.	<pre><ListBox x:Name="listBox1"> <ListBox.ItemTemplate> <DataTemplate> <StackPanel Orientation="Vertical" > <TextBlock Text="{Binding numara}" /> <Image Source="ogrenci.jpg" Width="75" Height="75" /> <TextBlock Text="{Binding isimsoyisim}" /> <TextBlock Text="{Binding durum}" /> <Image Source="{Binding durumikon}" Width="20" Height="20" /> </StackPanel> </DataTemplate> </ListBox.ItemTemplate> </ListBox></pre>												
➤ Veri listesini “ListBox”üzerinde listelemek için “MainPage()”metodunda gerekli kodları yazınız.	<pre>public MainPage() { InitializeComponent(); listBox1.ItemsSource = ogrencilisteal(); }</pre>												
➤ Uygulamayı çalıştırınız(F5). Sonuçları gözlemleyiniz.													

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. ItemsControl kullandınız mı?		
2. DataGrid kontrolünü kullandınız mı?		
3. ListBox kontrolünü kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. ItemsControl nesnesinin XAML yapısında liste eleman şablonunu tasarladığımız etiket aşağıdakilerden hangisidir?
A) ItemTemplate
B) DataTemplate
C) ItemsPanel
D) Template
2. ItemsControl üzerinde listelenecek kayıtları içeren veri kaynağını ItemsControl nesnesinin hangi özelliği ile ilişkilendirmemiz gerekir?
A) ItemTemplate
B) ItemsSource
C) Items
D) DataTemplate
3. Satır ve sütun yapısına sahip olan ve kayıtları tablo hâlinde gösteren listeleme kontrolü aşağıdakilerden hangisidir?
A) Canvas
B) ItemsContro
C) DataGrid
D) ListBox
4. Aşağıdakilerden hangisi DataGrid üzerinde oluşturabildiğimiz sütun türlerinden birisi **değildir**?
A) DataGridTextColumn
B) DataGridCheckBoxColumn
C) DataGridTemplateColumn
D) DataGridImageColumn
5. Liste kutusu(ListBox) elemanlarının tutulduğu dizi aşağıdakilerden hangisidir?
A) Items
B) Elements
C) Index
D) Objects

6. Liste elemanın seçili olup olmadığını belirlemek için hangi özelliğini kullanmalıyız?
- A) IsEnabled
 - B) IsSelected
 - C) Checked
 - D) Content

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-4

AMAÇ

Çoklu ortam kontrollerini kullanabileceksiniz.

ARAŞTIRMA

- “MediaElement” kontrolünü araştırınız.
- “VideoBrush” fırça yöntemini araştırınız.

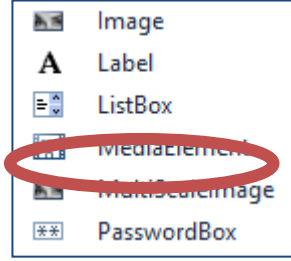
4. ÇOKLU ORTAM KONTROLLERİ

4.1. MediaElement Kontrolü

“MediaElement”, sayfa üzerindeses ve video dosyalarını yürütmemizi sağlayan kontroldür.



Resim 4.1: MediaElement kullanımı



Resim 4.2: ToolBox üzerinde MediaElement kontrolü

“MediaElement” kontrolünün bazı önemli özellikleri şunlardır:

Özellik	Görevi	Açıklama
Source	“MediaElement”in yürüteceği video ya da ses dosyasının ismini tutar.	“String” değer alır.
Volume	Ses seviyesi değerini ayarlar.	Sayısal değer alır.
IsMuted	Sesikapalı ya da açıkyapabilir.	Sesin kapalı olması için “True”, açık olması için “False” değeri alır.
Stretch	Video görselinin “MediaElement” içerisine nasıl yerleşeceğini belirlemek için kullanılır.	“Fill”, “UniForm” ya da “UniFormToFill” değerlerinden birini alır.
AutoPlay	Sayfa açıldığında içeriğin otomatik olarak yürütülmeye başlayıp başlamamasını belirler.	Otomatik başlaması için “True”, başlamaması için “False” değeri alır.
Balance	Sesin sağ ya da sol hoparlör dengesini ayarlar.	1 ve -1 arasında değerler alır.
Position	İçeriğin yürütülmesi sırasında, zaman ilerlemesinin değerini elde etmek veya belirli bir değere ayarlamak için kullanılır.	“TimeSpan” türünde değer alır.
BufferingProgress	Video yürütülürken arabelleğe alma ilerlemesinin yüzde değerini temsil eden bir sayı içerir.	0-1 arası bir değer içerir.
DownloadProgress	Video yürütülürken indirme yüzdesinin ilerlemesini temsil eden bir sayı içerir.	0-1 arası bir değer içerir.
NaturalDuration	“MediaElement” üzerinde yüklü olan videonun toplam süresini tutar.	“TimeSpan” türünde değer alır.

Tablo 4.1: “MediaElement” özellikleri

Not: Video dosyasını, projenin “Web” tarafında bulunan “ClientBin” klasörüne dâhil etmemiz gerekiyor. Aksi halde video verisini “xap” dosyasının içerisine entegre etmek durumunda kalırız. Bu durum “xap” dosyasının boyutunu artırır. Bu ise “xap” dosyasını kullanan web sayfasının, internet ortamında geç yüklenmesine sebep olur

Örnek: “MediaElement” için en temel özellikleri kullanan bir örneği inceleyelim.

```
<MediaElement x:Name="media1" Source="dersvideo.wmv" Balance="0"
Volume="5.6" AutoPlay="True" IsMuted="False"/>
```

4.2. MediaElement Kontrolünü Programlama

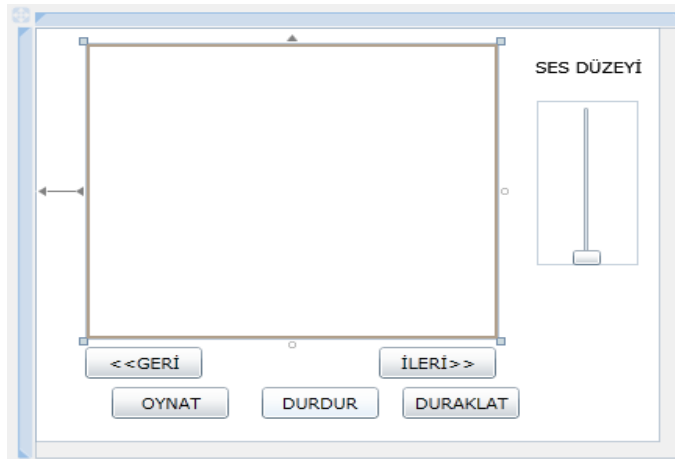
“MediaElement” in bazı metot ve özellikleri sayesinde, video yada ses dosyası yürütmesini kontrol edebiliriz. Bu metotlardan bazıları şunlardır:

Metod Adı	Görevi
Play()	Video ya da ses dosyası yürütmesini başlatır.
Stop()	Video ya da ses dosyası yürütmesini durdurur.
Pause()	Video ya da ses dosyası yürütmesini duraklatır.

Tablo 4.2: Bazı “MediaElement” metodları

Örnek: Basit bir video oynatıcı(media player)örneğini inceleyelim.

➤ Tasarım görüntüsü



Resim 4.3: Tasarım görüntüsü

➤ Program kodları

```
void btnbaslat_Click(object sender, RoutedEventArgs e)
{
    mediaElement1.Play(); //Videoyu oynatmaya başlar.
}

void btndurdur_Click(object sender, RoutedEventArgs e)
{
    mediaElement1.Stop(); //Videoyu durdurur.
}

void btnduraklat_Click(object sender, RoutedEventArgs e)
{
    mediaElement1.Pause(); //Videoyu duraklatır.
}

void btnileri_Click(object sender, RoutedEventArgs e)
{
    //Video oynarken 3 saniye ilerletir.
    TimeSpan sure = new TimeSpan(0, 0, 3);
    mediaElement1.Position = mediaElement1.Position.Add(sure);
}

void btngeri_Click(object sender, RoutedEventArgs e)
{
    //Videoyu 3 saniye geri sarar
    TimeSpan sure = new TimeSpan(0, 0, 3);
    mediaElement1.Position = mediaElement1.Position.Subtract(sure);
}
```

```

void slider1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
{
    //ses deęerini, slider nesnesine g re ayarlar.
    mediaElement1.Volume = slider1.Value;
}

```

➤ **Çalışma anı görüntüsü**



Resim 4.4: Çalışma anı görüntüsü

“MediaElement” i programlarken kullandığımız en  nemli olaylar ise şunlardır:

Olay Adı	G�revi
BufferingProgressChanged	Y�r�t�len videonun, �nbelleęe alınması(buffering) sırasında, �nbelleęe alma miktarının her deęişiminde tetiklenen olaydır.
DownloadProgressChanged	Y�r�t�len videonun indirilmesi sırasında, indirme miktarının her deęişiminde tetiklenen olaydır.
MediaOpened	Video ya da ses dosyasının y�r�t�lmesi bařladıęında tetiklenir.
MediaEnded	Video ya da ses dosyasının y�r�t�lmesi durduęunda tetiklenir.
MediaFailed	Video ya da ses dosyasının y�r�t�lmesi olaęan dıřı bi�imde(bir hata sebebiyle) durduęunda tetiklenir.
CurrentStateChanged	Oynatma, duraklatma, durdurma gibi durum deęişiklikleri meydana geldięinde tetiklenir.

Tablo 4.3: Bazı “MediaElement” olayları

4.3.VideoBrush

Bir alanı bir video içeriği ile boyamak için kullanılır. Kullanılacak video içeriği bir “MediaElement” tarafından sağlanır. “VideoBrush” geometrik şekillerin yanı sıra birçok kontrollere de uygulanabilir.

Örnek:

```
<Canvas

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<MediaElement

x:Name="MediaElement1"

Source="Wildlife.wmv" IsMuted="True"

Opacity="0.0" IsHitTestVisible="False" />

<TextBlockCanvas.Left="5" Canvas.Top="30"

FontFamily="Verdana" FontSize="120"

FontWeight="Bold" TextWrapping="Wrap"

Text="Video">

<TextBlock.Foreground>

<VideoBrush SourceName="MediaElement1" Stretch="UniformToFill" />

</TextBlock.Foreground>

</TextBlock>

</Canvas>
```

➤ Çalışma anı görüntüsü



Resim 4.5 : VideoBrushkullanımı

UYGULAMA FAALİYETİ

Bu uygulama faaliyetinde “MediaElement” kullanarak basit bir “player” yapacağız. Açıklamaları dikkate alarak uygulama faaliyetini gerçekleştiriniz.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir proje oluşturunuz.	➤ File→New→Project
➤ Kullanacağımız simgeleri ve video dosyasını “ClientBin” klasörüne kopyalayınız.	➤ İkonlar:  ➤ Video dosyası olarak “Belgelerim” altındaki “Wildlife.wmv” dosyasını kullanabilirsiniz.
➤ Bir “Rectangle” nesnesi ekleyiniz.	➤ Özelliklerini: RadiusX =”20”, RadiusY=”20”, StrokeThickness=”3”, Stroke=”Red” olarak ayarlayınız.
➤ Üç adet “Button” ekleyip XAML yapılarını düzenleyerek üzerlerinde başlat, durdur, duraklat ikonlarının görünmesini sağlayınız. 	<pre> <Button Content="Başlat" Height="40" Margin="161,163,199,97" Name="button1" Width="40"> <Button.ContentTemplate> <DataTemplate> <Image Source="play.jpg" Stretch="Fill"/> </DataTemplate> </Button.ContentTemplate> </Button> <Button Height="40" Margin="94,163,266,97" Name="button2" Width="40"> <Button.ContentTemplate> <DataTemplate> <Image Source="pause.jpg" Stretch="Fill"/> </DataTemplate> </Button.ContentTemplate> </Button> <Button Content="Durdur" Height="40" Margin="225,163,135,97" Name="button3" Width="40"> </pre>

	<pre> <Button.ContentTemplate> <DataTemplate> <Image Source="stop.jpg" Stretch="Fill"/> </DataTemplate> </Button.ContentTemplate> </Button> </pre>
➤ “MediaElement” ekleyiniz.	➤ Video kaynak dosyasını belirtiniz. (Source = “Wildlife.wmv”) ➤ “MediaElement”i görünmez duruma getiriniz.(Visible=”Collapsed”) ➤ Otomatik olarak başlamamasını sağlayınız.(AutoPlay=”False”) ➤ “MediaElement”in isminin verilmiş olmasına dikkat ediniz.

➤ “Rectangle” kontrolünü “VideoBrush” fırçası ile boyayıp videonun “Rectangle” üzerinde oynatılmasını sağlayınız.	<pre><Rectangle Height="145" Margin="94,22,106,178" Name="rectangle1" Stroke="Red" StrokeThickness="3" Width="286" RadiusX="20" RadiusY="20"> <Rectangle.Fill> <VideoBrush SourceName="mediaElement1" Stretch="UniformToFill" /> </Rectangle.Fill> </Rectangle></pre>
➤ Bir “Slider” ekleyiniz.	➤ Özelliklerini: Minimum = “0” ve Maximum = “1” yapınız.
➤ “MediaElement” in ses düzeyini değiştiren kodları “Slider” ın “ValueChanged” olayı içine yazınız.	<pre>private void slider1_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e) { mediaElement1.Volume = slider1.Value; }</pre>
➤ “Başlat”, “Durdur” ve “Duraklat” düğmelerinin “Click” olaylarına gerekli kodları yazınız.	<pre>private void button1_Click(...) { mediaElement1.Play(); } private void button2_Click(...) { mediaElement1.Pause(); } private void button3_Click(...) { mediaElement1.Stop(); }</pre>
➤ Uygulamayı çalıştırınız(F5). Sonuçları gözlemleyiniz.	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. “MediaElement” kontrolünü kullandınız mı?		
2. “MediaElement” programlamasını gerçekleştirdiniz mi?		
3. “VideoBrush” fırçasını kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdaki “MediaElement” özelliklerinden hangisi, oynatılan ses dosyası için hoparlör dengesini ayarlamayı sağlar?
A) Volume
B) Stretch
C) Balance
D) Source
2. “MediaElement” üzerinde yüklü olan videonun toplam süresini tutan özelliği aşağıdakilerden hangisidir?
A) Balance
B) NaturalDuration
C) Position
D) Length
3. “MediaElement” üzerinde yürütülen videoyu duraklatmayı sağlayan metot aşağıdakilerden hangisidir?
A) Play
B) Pause
C) Stop
D) End
4. “MediaElement” üzerinde video yürütülürken hata oluştuğunda tetiklenen olay aşağıdakilerden hangisidir?
A) MediaOpened
B) MediaEnded
C) MediaFailed
D) CurrentStateChanged

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

5. “MediaElement” kontrolünün ses seviyesi değerini ayarlayan özelliği.....dir.
6. “MediaElement” kontrolünde, sesin açık olup olmayacağını ayarlamak içinözellği kullanılır.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-5

AMAÇ

Veri bağlama işlemlerini gerçekleştirebileceksiniz.

ARAŞTIRMA

- Veri bağlama kurallarını araştırınız.
- DataContext özelliğini araştırınız.
- Liste bağlama(List Binding) işlemini araştırınız.
- Veri şablonlarını(DataTemplate) araştırınız.
- Master/detail veri bağlama yöntemini araştırınız.

2. BAĞLAMA İŞLEMLERİ

5.1.Bağlama Kuralları

XAML tabanlı internet uygulamaları geliştirmede veriye erişim büyük bir önem taşımaktadır. Veri bağlama(databinding) yöntemlerini kullanarak değişik veri kaynaklarından alabileceğimiz verileri web sayfalarında rahatlıkla kullanabiliriz. Ayrıca uygulamada kullandığımız farklı kontrollerin özelliklerini karşılıklı olarak birbirine bağlayıp program kodu yazmadan kontrollerin birbirleriyle etkileşimde bulunmasını sağlayabiliriz.

Veri bağlama(databinding) hakkında genel bir tanımlama yapacak olursak "Bir sınıfın herhangi bir üyesinin başka bir sınıfın bir üyesinin sahip olduğu veriye otomatik olarak erişmesidir." diyebiliriz.

- XAML kodlarıyla bir kontrolün bir özelliğini diğer bir kontrolün veya veri tipinin bir özelliğine bağlamak için {Binding} bildirisini kullanırız.
- Bildirim kuralı küme parantezleri içinde "Binding" anahtar kelimesini izleyen ve virgülle ayıracağımız bir dizi özelliğe değer atamaktan ibarettir. "Binding" bildirisinin yazım yöntemi aşağıdaki gibidir.

{BindingPath=Özellik,Mode=Veri Akış Yönü,ElementName=Kontrol Adı,
Source=Veri Kaynağı}

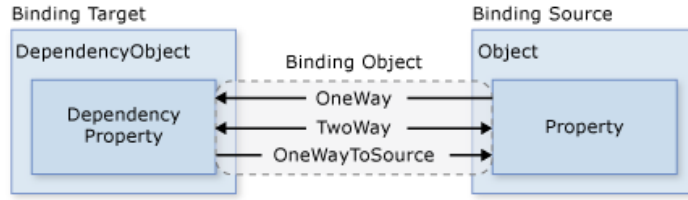
Örnek kullanım

```
<TextBox Name="textBox1"/>  
<TextBlock Text="{Binding Path=Text, Mode=OneWay, ElementName=textBox1}" />
```

Resim 5.1: “Binding” kullanımı

“Binding” bildirisinin dört temel bileşeni vardır. Bunlar:

- Path: Kaynak nesnenin, bağlanıp veriyi alacağımız özelliğinin adını tutar.
- ElementName: Veriyi alacağımız kaynak nesne eğer bir XAML kontrolü ise “ElementName” bileşenine kontrolün ismini atamamız gerekir.
- Source: Veriyi alacağımız kaynak, XAML kontrolü olmayan bir nesne ise “Source” özelliğine bu nesnenin ismi atanır.
- Mode: Nesnelerin birbirine bağladığımız özellikleri arasında veri akış yönünü belirlemek için kullanılır. “Mode” bileşenine atayabileceğimiz bazı değerler sayesinde veri aktarım yönünü belirleriz. Bu değerler şunlardır:
 - **OneWay:** Kaynaktan hedefe tek yönlü bağlantıdır. Yani kaynak nesnenin özelliği değiştirildiğinde hedef nesnenin özelliği de değişir; fakat hedef nesnenin özelliğinin değiştirilmesi kaynak nesnenin özelliğini etkilemez.
 - **OneWayToSource:** OneWay bağlamanın tersidir. Hedeften kaynağa tek yönlü bağlantıdır. Yani hedef nesnenin özelliği değiştirildiğinde kaynak nesnenin özelliği de değişir; fakat kaynak nesnenin özelliğinin değiştirilmesi hedef nesnenin özelliğini etkilemez.
 - **TwoWay:** Çift yönlü bağlantı oluşturur. Yani hedef ve kaynak nesnelerin birbirine bağlanan özellikleri karşılıklı olarak birbirine veri aktarabilir.
 - **OneTime:** Hedef nesnenin özelliğinin değeri, uygulama başladığında bir defaya mahsus olarak kaynak nesnenin özelliğinden alınır.



Resim 5.2 : Bağlama(Binding) işleminde veri akış yönleri

XAML kontrolleri arasında veri bağlamayla ilgili birkaç örneğini inceleyelim.

Örnek: “MediaElement” üzerinde bir video oynatılırken video ilerlemesine bağlı olarak “Slider” sürgüsünün otomatik olarak ilerlemesini sağlayalım.

```
<StackPanel>

<MediaElement x:Name="media1" Source="Wildlife.wmv" Width="200" Height="200" />

<Slider Value="{Binding Path=Position.TotalSeconds, Mode=OneWay,
ElementName=media1}" Width="200" x:Name="slider1"/>

</StackPanel>
```

Burada kaynak“MediaElement”, hedef ise“Slider”dir. “MediaElement”in “Position.TotalSeconds”özelliliği ile “Slider”in “Value”özelliliği arasında tek yönlü bağ kurulduğu görülmektedir.

Value = "{Binding Path=Position.TotalSeconds, Mode=OneWay, ElementName=media1}"

Yapılan “Binding” bildirisiyle ilgili olarak şunları söyleyebiliriz:

- “Value” özelliğinde {Binding} bildirisi oluşturmak “Value”nin hedef olmasını ve değerinin dış bir kaynaktan alınmasını sağlar.
- BildirininPath=Position.TotalSecondskısmı ise“MediaElement”üzerindeki videonun ilerleme pozisyonunu tutan “Position.TotalSeconds” özelliğinin“Slider”in “Value” özelliğine kaynaklık ederekveri göndermesini sağlar.
- **Mode=OneWay** kısmında bağlama tek yönlü(“MediaElement” kontrolünden“Slider” kontrolüne) olarak kurulmuştur.
- **ElementName=media1**kısımında kaynak kontrololan “MediaElement”in ismi belirtilmiştir.

Bu örnekte “Slider”in“Maximum” değerini oynatılan video dosyasının toplam süresine(NaturalDuration.TimeSpan.TotalSeconds) eşitlememiz gerekiyor. Onun için “MediaElement”in “MediaOpened” olayına aşağıdaki kodu yazacağız.

```
privatevoid media1_MediaOpened(object sender, RoutedEventArgs e)
{
```

➤ **Çalışma anı görüntüsü**



Resim 5.3: “MediaElement”ile “Slider” bağlantısı

Örnek: İki “TextBox” kontrolünün “Text” özelliklerini çift yönlü bağlama (TwoWay) yöntemiyle bağlayalım.

```
<StackPanel>

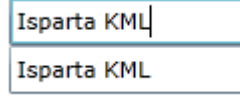
<TextBox x:Name="text1" Width="120" />

<TextBox x:Name="text2" Text="{Binding Path=Text, Mode=TwoWay,
ElementName=text1}" Width="120"/>

</StackPanel>
```

Kodlardan anlaşılacağı gibi kaynak kontrol “text1” ve hedef kontrol “text2”dir.

➤ Çalışma anı görüntüsü



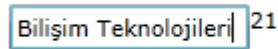
Resim 5.4: İki “TextBox”u çift yönlü olarak bağlama

Örnek: “TextBox” kontrolüne metin girişi sırasında girilen karakter sayısını bir “TextBlock” üzerinde gösteren bir örneği inceleyelim.

```
<Grid x:Name="LayoutRoot" Background="White">  
  
<StackPanel Orientation="Horizontal" Margin="50">  
  
<TextBox Height="23" Name="textBox1" Width="120" />
```

Görüldüğü gibi bir “TextBox”un Length özelliği diğer “TextBox”un Text özelliğine tek yönlü olarak bağlanmıştır.

➤ Çalışma anı görüntüsü



Resim 5.5: Çalışma anı Görüntüsü

5.2. Veri Bağlama(DataContext)

XAML kontrollerinde bulunan“DataContext” özelliği, kontrolün dış kaynaklı bir veriye bağlanması için kullanılır. Veri, kontrolün hangi özelliğine bağlanacaksa o özelliğe {Binding} bildirisi oluşturmamız gerekir. Örneğin:

```
<TextBox Height="23" Name="textBox1" Width="120"Text="{Binding}" />
```

“Text” özelliğine {Binding} bildirisi oluşturarak “Text” özelliğinin dış bir veriye bağlanacağını belirtmiş oluruz. Ancak “TextBox” kontrolünü veriye bağlamak için “DataContext” özelliğini kullanmamız gerekir.

```
public MainPage()
{
    InitializeComponent();
    textBox1.DataContext = "Merhaba";
}
```

Bu şekilde“TextBox” üzerinde “Merhaba” ifadesi yazdırılmış olur.

Yukarıda “DataContext” özelliğinin en basit anlamda kullanımı gösterilmiştir. Bununla beraber “DataContext” özelliği, birkaç alandan oluşan bir kayıt yapısını bir kontrol kümesine bağlamak için de kullanılır.Örneğin bir “Canvas”ın “DataContext” özelliğine bir kayıt aktarıldığında,“Canvas” içinde bulunan “TextBlock” gibi alt kontroller,kayıt yapısının içindeki farklı veri alanlarına,{Binding} yöntemiyle ulaşabilirler.

Örnek:“DataContext” kullanarak birkaç özelliği bulunan bir nesneyi bir kontrol kümesine bağlayalım.

➤ “Personel” isminde bir sınıf(class) yapısı tanımlayacağız.

```
publicclassPersonel
{
    publicstring tckimlik { get; set; }
    publicstring adsoyad { get; set; }
    publicstring adres { get; set; }
}
```

- Kimlik no, ad-soyad ve adres bilgilerini gösterecek olan üç adet “TextBlock” kontrolünü bir “StackPanel” içine yerleştirip veri bağlantılarını kuracağız.

```
<StackPanel Orientation="Vertical" Margin="50" x:Name="stackPanel1">
<TextBlock Name="textBlock1"Text="{Binding tckimlik}" />
<TextBlock Name="textBlock2"Text="{Binding adsoyad}" />
<TextBlock Name="textBlock3"Text="{Binding adres}" />
</StackPanel>
```

- Sayfa açılışında “Personel” sınıfından bir nesne örnekleyerek, oluşturulan nesnenin “tckimlik”, “adsoyad” ve “adres” özelliklerine değerler atayacağız.

```
public MainPage()
{
    InitializeComponent();

    Personel yenipersonel = newPersonel { tckimlik = "51978438593", adsoyad
= "İbrahim GÜNEŞ", adres = "Bahçelievler Mahallesi" };
}
```

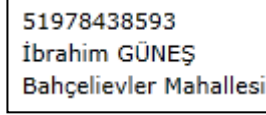
- Oluşturduğumuz “yenipersonel” isimli nesneyi “StackPanel”in “DataContext” özelliğine aktaracağız. Bu sayede “StackPanel” içinde bulunan ve veri bağlantıları yapılan“TextBlock” kontrolleri üzerinde kimlik no, ad-soyad ve adres bilgileri görüntülenmiş olur.

```
public MainPage()
{
    InitializeComponent();

    Personel yenipersonel = newPersonel { tckimlik = "51978438593", adsoyad
= "İbrahim GÜNEŞ", adres = "Bahçelievler Mahallesi" };

    stackPanel1.DataContext = yenipersonel;
}
```

➤ Çalışma anı görüntüsü



Resim 5.6: Çalışma anı görüntüsü

5.3.Liste Bağlama(List Binding)

XAML tabanlı internet uygulamalarında diziler gibi liste hâlindeki verileri kolaylıkla “ListBox”, “ItemsControl” ve “DataGrid” gibi listeleme kontrollerine aktarıp listeleyebiliriz. Hatta listeleme kontrollerinde gerekli şablon tasarımı yapılarak görsel ve kullanışlı listeleme kontrolleri hazırlayabiliriz.

“ListBox” vb. listeleme kontrollerinin “ItemsSource” özelliğine doğrudan doğruya bir veri dizisini aktarabiliriz. Buna alternatif olarak “ItemsSource” özelliğinde {Binding} bildirisi oluşturularak kontrolün liste elemanlarını “DataContext” özelliği üzerinden transfer etmesini sağlayabiliriz. Konu ile ilgili olarak basit bir örneği inceleyelim.

Örnek: Bir diziyi bir “ListBox” kontrolüne bağlayalım.

➤ XAML yapısı

```
<Grid x:Name="LayoutRoot" Background="White">
<ListBox x:Name="listBox1" ItemsSource="{Binding}" Width="140" Height="80"
Margin="30" />
</Grid>
```

İşaretli kısma dikkat edelim. “ItemsSource” özelliğinde {Binding} bildirisi kullanılarak liste elemanlarının “DataContext” özelliğine aktarılan veri kümesinden alınacağı belirtilmiş olur.

➤ **Program kodları**

```
public MainPage()
{
    InitializeComponent();

    string[] iller = newstring[5]
        {"ISPARTA", "BURDUR", "ANTALYA", "MERSİN", "ADANA"};

    listBox1.DataContext = iller;
}
```

İl isimlerinden oluşan beş elemanlı bir “string” dizisi “ListBox” kontrolünün “DataContext” özelliğine aktarılmaktadır.

➤ **Çalışma anı görüntüsü**



Resim 5.7: Çalışma anı görüntüsü

Örnek 2: Bir sınıf yapısından ürettiğimiz listeyi “DataGrid” kontrolüne bağlayalım.

- Şehirler için isim, plaka kodu ve bölge bilgilerini içeren “Şehir” isimli bir sınıf tanımlayacağız.

```
publicclassŞehir
{
    publicstring isim { get; set; }

    publicstring bolge { get; set; }

    publicint plakakod { get; set; }
}
```

- “Şehir” sınıfından nesnelistesi oluşturup geri döndüren “ŞehirListeGetir()” isimli metodu yazacağız.

```
public List<Şehir> ŞehirListeGetir()
{
    List<Şehir> Şehirler = newList<Şehir>(){
        newŞehir{ bolge="Marmara", isim="İstanbul", plakakod=34},
        newŞehir{ bolge="Ege", isim="İzmir", plakakod=35},
        newŞehir{ bolge="İç Anadolu", isim="Ankara", plakakod=6},
        newŞehir{ bolge="Doğu Anadolu", isim="Ağrı", plakakod=4},
        newŞehir{ bolge="İç Anadolu", isim="Kayseri", plakakod=38},
        newŞehir{ bolge="Akdeniz", isim="Isparta", plakakod=32}
    };
    return Şehirler;
}
```

Sayfada bir “DataGrid” ve bir “Button” kullanacağız. “DataGrid” sütunlarını, “Binding” yöntemiyle, “Şehir” sınıfının ilgili özelliklerine bağlayacağız.

```
<StackPanel x:Name="LayoutRoot" Background="White" Margin="40">

<my:DataGrid AutoGenerateColumns="False" Height="Auto" Name="dataGrid1"
Width="300" ItemsSource="{Binding}">

<my:DataGrid.Columns>

<my:DataGridTextColumn Header="ŞEHİR ADI" Binding="{Binding isim}"/>

<my:DataGridTextColumn Header="PLAKA KODU" Binding="{Binding plakakod}"/>

<my:DataGridTextColumn Header="BÖLGE" Binding="{Binding bolge}"/>

</my:DataGrid.Columns>

</my:DataGrid>

<Button Content="YÜKLE" Width="200" Click="Button_Click"></Button>

</StackPanel>
```

- “ŞehirListeGetir()” metodunun oluşturduğu listeyi “DataGrid”in “DataContext” özelliğine aktaran program kodunu düğmenin “Click” olayındayazacağız.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    .....
}
```

- Çalışma anı görüntüsü

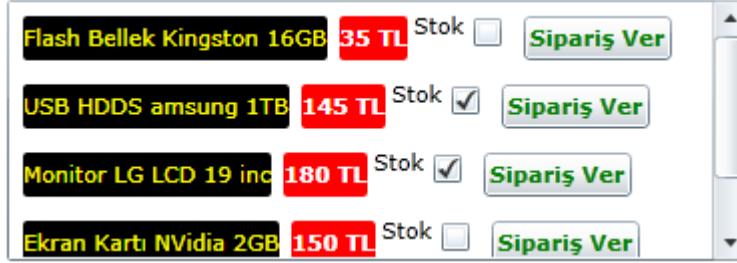
ŞEHİR ADI	PLAKA KODU	BÖLGE
İstanbul	34	Marmara
İzmir	35	Ege
Ankara	6	İç Anadolu
Ağrı	4	Doğu Anadolu
Kayseri	38	İç Anadolu
Isparta	32	Akdeniz

YÜKLE

Resim 5.8: Çalışma anı görüntüsü

5.4. Veri Şablonu(DataTemplate)

“ListBox”, “ComboBox”, “ItemsControl” ve “DataGrid” gibi listeleme kontrollerini kullanırken bazen her bir liste elemanında birden fazla bileşeni barındırmamız gerekebilir. Aşağıdaki resmi inceleyiniz.



Resim 5.9: “DataTemplate” uygulanmış bir “ListBox”

Liste elemanlarının, ürün adı, ürün fiyatı, stok durumu bilgilerini temsil eden birkaç kontrol ve “Sipariş Ver” etiketli bir düğme barındırdığını görmekteyiz.

Dış kaynaklı bir kayıt listesini “ListBox” vb. kontrollerde listelerken liste elemanları içinde, kayıt yapısındaki veri alanlarının birkaçını ya da tümünü görüntülemek istediğimizde veri şablonu(DataTemplate) oluşturmamız gerekir. Örneğin ürün adı, fiyatı, stok durumu gibi özellikler içeren bir sınıftan üretilmiş yirmi elemanlı bir listenin bulunduğunu düşünelim. Bu listeyi bir “ListBox” kontrolüne bağladığımızda her bir liste elemanı üzerinde her bir kayda ait ürün adı, fiyatı ve stok durumu bilgilerini ayrı ayrı görüntüleyebilmek için “ListBox” kontrolünün liste elemanı şablonunu tasarlamamız gerekir. Bunun için `<DataTemplate></DataTemplate>` yapısını kullanırız.

Liste elemanlarının görüntüleyeceği kayıt yapısının içindeki veri alanlarını temsil edecek kontroller grubu `<DataTemplate></DataTemplate>` yapısı içine eklenir. “DataTemplate” yapısı içine eklediğimiz kontroller “Binding” bildirileriyle farklı veri alanlarına bağlanırlar.

Bir örnek uygulama üzerinden ilerleyerek konuyu kavramaya çalışalım.

- Ürün adı(isim), fiyatı (fiyat) ve stok durumu (stok) bilgilerini içeren “Urun” isminde bir sınıf tanımlayacağız.

```
publicclassUrun
{
    publicstring isim { get; set; }
    publicstring fiyat { get; set; }
    publicbool stok { get; set; }
}
```

- “Urun” sınıfından bir liste oluşturan “UrunListeGetir()” ismindeki metodu oluşturacağız.

```
public List<Urun> UrunListeGetir()
{
    List<Urun> Urunler = newList<Urun>(){
        newUrun{ isim="Flash Bellek Sony 16GB", fiyat="35 TL", stok=false},
        newUrun{ isim="USB HDDS amsung 1TB", fiyat="145 TL", stok=true},
        newUrun{ isim="Monitor LG LCD 19 inc", fiyat="180 TL", stok=true},
        newUrun{ isim="Ekran Kartı NVidia 2GB", fiyat="150 TL", stok=false}
    };
    return Urunler;
}
```

- “ListBox” kontrolünü ve eleman şablonunu tasarlayacağız. Ayrıca ürün listesini oluşturacak bir “Button” da ekleyeceğiz.

```
<StackPanel x:Name="LayoutRoot" Background="White" Margin="40">

<ListBox Height="130" Name="listBox1" Width="370" ItemsSource="{Binding}">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<dataInput:Label Content="{Binding isim}" Background="Black" Foreground="Yellow"
Margin="3" />

<dataInput:Label Content="{Binding fiyat}" Background="Red" Foreground="White"
FontWeight="Bold" Margin="3" />

<TextBlock Text="Stok" />

<CheckBox IsChecked="{Binding stok}" Margin="3"/>

<Button Content="Sipariş Ver" Foreground="Green" FontWeight="Bold" Margin="3"
IsEnabled="{Binding stok}"

/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Button Content="Kayıtları Getir" Width="200" Margin="3" Click="Button_Click"
/>

</StackPanel>
```

- `<DataTemplate>` bloğunun `<ListBox.ItemTemplate>` bloğu içinde oluşturulduğuna dikkat edelim.
- Ayrıca `<DataTemplate>` bloğu içinde kontrollerin yatay yönlendirilmiş bir `StackPanel` içine yerleştirildiğini görmekteyiz.
- Veri şablonu içinde `Binding` yöntemiyle `Urun` sınıfının `isim` alanı bir `Label` kontrolüne, `fiyat` alanı diğer bir `Label` kontrolüne ve `stok` alanı bir `CheckBox` kontrolüne bağlanmıştır.
- Bazı kontrollerin `BackGround`, `Foreground` gibi renklendirme özellikleri de kullanılmıştır.
- Ayrıca şablon içinde `Sipariş Ver` etiketli düğme için ilginç bir bağlama yöntemi kullanılmıştır. Aşağıdaki koda dikkat edelim.

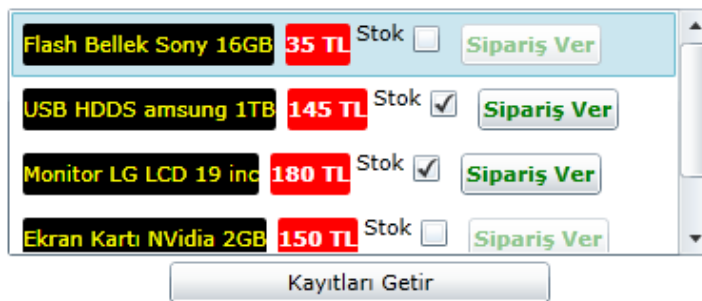
```
<Button Content="Sipariş Ver" Foreground="Green" FontWeight="Bold"
Margin="3" IsEnabled="{Binding stok}"
/>
```

Stokta olmayan ürünler için `Sipariş Ver` düğmesinin aktif olmamasını sağlamak amacıyla düğmenin aktif olma(`IsEnabled`) özelliği, `Urun` sınıfının `stok` alanına bağlanmıştır.

`Urun` sınıfından bir liste üretip bu listeyi `ListBox` kontrolüne bağlayan program kodunu `Button` kontrolünün `Click` olayına yazacağız.

```
privatevoid Button_Click(object sender, RoutedEventArgs e)
{
```

➤ Çalışma anı görüntüsü



Resim 5.10: Çalışma anı görüntüsü

5.5. Master/Detail Data Binding

“ListBox”(v.b. kontroller) üzerinde ana bilgileri listelenmiş kayıtlardan birisi seçildiğinde bu kaydın detay bilgilerinin “ListBox” haricinde bir grup kontrol üzerinde görüntülenmesi esasına dayanan uygulamalar geliştirirken ana/detay veri bağlama(master/detail data binding) yöntemini kullanabiliriz. Aşağıdaki resimde bu yöntemle çalışan bir uygulamanın görüntüsü bulunmaktadır.

Resim 5.11:Bağlama(master/detail data binding) yöntemi görüntüsü

Resmi incelediğimizde,“ListBox”tan bir ürünün ismi seçildiğinde bu ürünün ismiyle beraber, fiyat, stok durumu, fotoğraf gibi diğer verilerinin üç“TextBox” ve bir “Image” üzerinde görüntülediği” anlamaktayız. Buna benzer bir uygulama üzerinden ilerleyelim.

➤ **“Urun” isminde bir sınıf yapısı oluşturacağız.**

```
publicclassUrun
{
    publicstring isim { get; set; }
    publicstring resim { get; set; }
    publicstring stok { get; set; }
    publicstring fiyat { get; set; }
}
```

- “Urun” sınıfından liste oluşturan metodu oluşturacağız.

```
public List<Urun> UrunListeGetir()
{
    List<Urun> Urunler = newList<Urun>(){
        newUrun{ isim="Acer E1531", fiyat="980 TL", stok="Stokta Var",
            resim="acer.jpg"},
        newUrun{ isim="Samsung ATIV 5", fiyat="1930 TL", stok="Stokta Var",
            resim="samsung.jpg"},
        newUrun{ isim="Casper CN 847B", fiyat="900 TL", stok="Stokta Yok",
            resim="casper.jpg"},
        newUrun{ isim="Asus N56VZ", fiyat="1945 TL", stok="Stokta Var",
            resim="asus.jpg"}
    };
    return Urunler;
}
```

- Sadece ürün isimlerinin listeleneceği bir “ListBox” yapısı oluşturacağız.

```
<Grid x:Name="LayoutRoot" Background="White" Width="700" Height="432">

<Canvas Height="294" HorizontalAlignment="Left" Margin="30,63,0,0"
Name="canvas1" VerticalAlignment="Top" Width="200">

<ListBoxCanvas.Left="59" Canvas.Top="65" Height="164" Name="listBox1"
Width="120" ItemsSource="{Binding}" SelectionChanged="listBox1_SelectionChanged">

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding isim}" />

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<TextBlockCanvas.Left="88" Canvas.Top="13" Height="23" Name="textBlock1"
Text="ÜRÜNLER" FontWeight="Bold" />

</Canvas>
```

- **Kodlarda işaretlenen kısımlara dikkat edelim.**

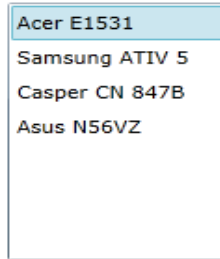
- “ListBox”un bir “Canvas” içinde yer alması sağlanmıştır.
- “ItemsSource” özelliği “DataContext” üzerinden gelecek listeye bağlanmıştır.
- “DataTemplate” içinde “Urun” sınıfının sadece “isim” özelliğine bağlanan tek bir “TextBlock” oluşturulmuştur. “ListBox”, “Urun” sınıfından üretilen listeye bağlandığında “ListBox” üzerindeki her bir liste elemanında “isim” özelliğinin yanı sıra “fiyat”, “stok”, “resim” özellikleri de bulunacaktır. Fakat sadece “isim” bilgileri görüntülenecektir.
- “ListBox”un “SelectionChanged” olayı da kullanılmıştır. Bu olay “ListBox” üzerinde bir eleman seçildiği anda çalışır.
- “Urun” sınıfından ürettiği listeyi “ListBox”a bağlayan program kodunu sayfa açılışında çalışan “MainPage()” metoduna yazacağız.

```
public MainPage()
{
    InitializeComponent();

    listBox1.DataContext = UrunListeGetir();
}
```

Bu aşamada uygulamayı çalıştırdığımızda sadece ürün isimlerinin “ListBox” üzerinde listelendiğini göreceğiz.

ÜRÜNLER



Resim 5.12: Ürünler

- Uygulamayı geliştirmeye devam edeceğiz. “ListBox”un yan tarafında ayrı bir “Canvas” oluşturacağız. Bu “Canvas” içinde ise “ListBox” tan seçilen ürünün detay bilgilerini üzerlerinde gösterecek olan kontroller grubunu oluşturup veri bağlantılarını yapacağız. Bu kontrollerin ayrı bir “Canvas” içinde olmaları gereklidir. Çünkü “ListBox” tan seçilen ve “isim”, “fiyat”, “stok” ve “resim” gibi bilgileri içeren elemanı “Canvas” kontrolünün “DataContext” özelliğine göndereceğiz.

-
-

././ 3. Adımda yazılan XAML kodlarının devamı

```
<Canvas Height="294" HorizontalAlignment="Left"
Margin="250,63,0,0" Name="canvas2" VerticalAlignment="Top" Width="321">

<TextBlockCanvas.Left="92" Canvas.Top="13" Height="23" Name="textBlock2"
Text="ÜRÜN DETAYI" FontWeight="Bold" />

<TextBlockCanvas.Left="198" Canvas.Top="42" Height="23" Name="textBlock3"
Text="RESİM:" FontWeight="Bold" />

<TextBlockCanvas.Left="20" Canvas.Top="93" Height="23" Name="textBlock4"
Text="FİYAT:" FontWeight="Bold" />

<TextBlockCanvas.Left="18" Canvas.Top="42" Height="23" Name="textBlock5"
Text="ÜRÜN ADI:" FontWeight="Bold" />

<TextBoxCanvas.Left="20" Canvas.Top="64" Height="23" Name="textBox1"
Width="149" Text="{Binding isim}"/>

<TextBoxCanvas.Left="18" Canvas.Top="113" Height="23" Name="textBox2"
Width="151" Text="{Binding fiyat}" />

<TextBlockCanvas.Left="18" Canvas.Top="142" Height="23" Name="textBlock6"
Text="STOK DURUMU:" FontWeight="Bold" />

<TextBoxCanvas.Left="18" Canvas.Top="162" Height="23" Name="textBox3"
Width="151" Text="{Binding stok}" />

<BorderCanvas.Left="191" Canvas.Top="64" BorderThickness="1"
BorderBrush="Black">

<Image Height="121" Name="image1" Stretch="Fill" Width="124" Source="{Binding
resim}" />

</Border>

<ButtonCanvas.Left="94" Canvas.Top="206" Content="SİPARİŞ VER" Height="34"
Name="button1" Width="120" FontWeight="Normal" />

</Canvas>

</Grid>
```

Görüldüğü gibi oluşturulan “Canvas” kontrolüne “canvas2” ismi verilmiş ve “Canvas” içinde oluşturulan “TextBox” kontrolleri ile “Image” kontrolü “Urun” sınıfının ilgili özelliklerine bağlanmıştır.

- Son olarak “ListBox” tan seçilen ürünün “isim”, “fiyat”, “stok” ve “resim” bilgilerini içeren “SelectedItem” özelliğini “canvas2” ismini verdiğimiz “Canvas” ın “DataContext” özelliğine aktaran program kodunu “ListBox” un “SelectionChanged” olayında yazacağız.

```
private void listBox1_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    canvas2.DataContext = listBox1.SelectedItem;
}
```

- Çalışma anı görüntüsü

The screenshot displays a web application interface with two main sections: "ÜRÜNLER" (Products) and "ÜRÜN DETAYI" (Product Details). In the "ÜRÜNLER" section, a list box contains four items: "Acer E1531", "Samsung ATIV 5", "Casper CN 847B", and "Asus N56VZ". The "Asus N56VZ" item is selected and highlighted. In the "ÜRÜN DETAYI" section, there are three input fields: "ÜRÜN ADI:" (Product Name) with the value "Asus N56VZ", "FİYAT:" (Price) with the value "1945 TL", and "STOK DURUMU:" (Stock Status) with the value "Stokta Var". To the right of these fields is a placeholder for the product image, labeled "RESİM:", which shows a laptop. At the bottom right of the details section is a button labeled "SİPARİŞ VER" (Place Order).

Resim 5.13:Çalışma anı görüntüsü

UYGULAMA FAALİYETİ

Bu uygulama faaliyetinde ana/detay veri bağlama(master/detail databinding) yöntemi kullanarak bir okulda bulunan sınıf isimlerinin liste kutusunda(ListBox) listelenmesini ve listeden seçilen bir sınıfa ait ayrıntılı bilgilerin “TextBox” kontrolleri üzerinde görüntülenmesini sağlayabiliriz.

Açıklamaları dikkate alarak uygulama faaliyetini gerçekleştiriniz.

İşlem Basamakları	Öneriler
➤ Tasarım editöründe yeni bir Silverlight projesi oluşturunuz.	➤ File→New→Project
➤ “Sınıf” isminde “Class” yapısı oluşturunuz.	<pre>publicclassSınıf {publicstring okul { get; set; } publicstring isim { get; set; } publicstring mevcut { get; set; } publicstring alani { get; set; } }</pre>
➤ Okuldaki sınıfların bilgilerini liste hâlinde oluşturan “SınıfBilgiGetir()” isimli metodu yazınız.	<pre>publicList<Sınıf> SınıfBilgiGetir() {List<Sınıf> Sınıflar = newList<Sınıf>(){ newSınıf{ isim="10/A", alani="Bilişim Teknolojileri", mevcut="8", okul="Anadolu Kız Meslek Lisesi"}, newSınıf{ isim="10/A", alani="Çocuk Gelişimi", mevcut="21", okul="Kız Meslek Lisesi"}, newSınıf{ isim="10/A", alani="Bilişim Teknolojileri", mevcut="17", okul="Kız Teknik Lisesi"}, newSınıf{ isim="10/B", alani="Çocuk Gelişimi", mevcut="27", okul="Kız Meslek Lisesi"}, newSınıf{ isim="11/A", alani="Bilişim Teknolojileri", mevcut="19", okul="Anadolu Kız Meslek Lisesi"} };return Sınıflar; }</pre>
➤ Sayfaya “ListBox” ekleyiniz.	➤ Araç kutusunu(ToolBox) kullanabilirsiniz.

➤ “ListBox” un eleman şablonunda sınıf ismi ve alanın yan yana bulunmasını sağlayacak düzenlemeyi yapınız. 10/A --- Bilişim Teknolojileri 10/A --- Çocuk Gelişimi 10/A --- Bilişim Teknolojileri 10/B --- Çocuk Gelişimi 11/A --- Bilişim Teknolojileri	<pre> <Grid x:Name="LayoutRoot" Background="White"> <ListBox Height="180" HorizontalAlignment="Left" Margin="42,44,0,0" Name="listBox1" VerticalAlignment="Top" Width="230" ItemsSource="{Binding}"> <ListBox.ItemTemplate> <DataTemplate> <StackPanel Orientation="Horizontal"> <TextBlock Text="{Binding isim}" Margin="5"/> <TextBlock Text="---" Margin="5" /> <TextBlock Text="{Binding alan}" Margin="5" /> </StackPanel> </DataTemplate> </ListBox.ItemTemplate> </ListBox> </Grid> </pre>
➤ Sayfa açılışında sınıfların listesinin oluşturulup “ListBox”a yüklenmesi için gerekli program kodlarını yazınız.	<pre> public MainPage() { InitializeComponent(); listBox1.DataContext = SinifBilgiGetir(); } </pre>
➤ Sayfaya “Canvas” ekleyerek içerisine “TextBox” ve “TextBlock” yerleştirip aşağıda görünen tasarımı yapınız.	➤ ToolBox kullanarak kontrolleri ekledikten sonra XAML tarafında bağlama işlemleri yapınız. <pre> <Canvas Height="180" HorizontalAlignment="Left" Margin="311,44,0,0" Name="canvas1" VerticalAlignment="Top" Width="263"> <TextBlock Canvas.Left="16" Canvas.Top="19" Height="23" Name="textBlock1" Text="SINIF ADI:" FontWeight="Bold" /> <TextBox Canvas.Left="96" Canvas.Top="18" Height="23" </pre>

SINIF ADI: OKULU: ALANI: MEVCUDU:	<pre> Name="textBox1" Width="150"Text="{Binding isim}" /> <TextBlockCanvas.Left="16" Canvas.Top="48" FontWeight="Bold" Height="23" Name="textBlock2" Text="OKULU:" /> <TextBoxCanvas.Left="96" Canvas.Top="48" Height="23" Name="textBox2" Width="150"Text="{Binding okul}" /> <TextBlockCanvas.Left="16" Canvas.Top="80" FontWeight="Bold" Height="23" Name="textBlock3" Text="ALANI:" /> <TextBoxCanvas.Left="96" Canvas.Top="80" Height="23" Name="textBox3" Width="150"Text="{Binding alani}" /> <TextBlockCanvas.Left="16" Canvas.Top="109" FontWeight="Bold" Height="23" Name="textBlock4" Text="MEVCUDU:" /> <TextBoxCanvas.Left="96" Canvas.Top="109" Height="23" Name="textBox4" Width="150"Text="{Binding mevcut}" /> </Canvas> </pre>
➤ “ListBox” tan seçilen bir sınıfın tüm bilgilerini “Canvas” a gönderen program kodunu yazınız.	<pre> privatevoid listBox1_SelectionChanged(object sender, SelectionChangedEventArgs e) { canvas1.DataContext = listBox1.SelectedItem; } </pre>
➤ Uygulamayı çalıştırıp(F5) sonuçları gözlemleyiniz.	ISPARTA KIZ TEKNİK VE MESLEK LİSESİ 10/A --- Bilişim Teknolojileri 10/A --- Çocuk Gelişimi 10/A --- Bilişim Teknolojileri 10/B --- Çocuk Gelişimi 11/A --- Bilişim Teknolojileri SINIF ADI: 10/A OKULU: Kız Teknik Lisesi ALANI: Bilişim Teknolojileri MEVCUDU: 17

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Kontrollerin “Binding” özelliğini kullandınız mı?		
2. “DataContext” özelliğini kullandınız mı?		
3. Liste bağlama işlemini yaptınız mı?		
4. Veri şablonu(DataTemplate) oluşturdunuz mu?		
5. Master/Detail veri bağlama işlemini yaptınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1.bir sınıfın herhangi bir üyesinin başka bir sınıfın bir üyesinin sahip olduğu veriye otomatik olarak erişmesi olarak tanımlanabilir.
2. XAML içerisinde bir kontrolün bir özelliğini diğer bir kontrolün veya veri tipinin bir özelliğine bağlamak için.....bildirisi kullanılmalıdır.
3. Binding bildirisinde.....özelligi kaynak nesnenin, bağlanıp veriyi alacağımız özelliğinin adını tutar.
4. Binding bildirisinde.....özelligi bağlanan nesnelerin ilişkilendirdiğimiz özellikleri arasında veri akış yönünü belirler.
5. XAML kontrollerinde bulunan.....özelligi kontrolün bir veriye bağlanması için kullanılır.
6. “ListBox” ta listelenecek her bir kaydın içerdiği verilere bağlanıp temsil eden kontroller grubu.....bloğu içinde oluşturulur.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise “Modül Değerlendirme” ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Metin kutusu(TextBox) içeriğinin her değişiminde tetiklenen olay aşağıdakilerden hangisidir?
A) Click
B) MouseEnter
C) TextChanged
D) Checked
2. Takvim(Calendar) kontrolü üzerinde bir tarih seçildiği anda aşağıdaki olaylardan hangisi çalışır?
A) SelectedDatesChanged
B) DisplayDateChanged
C) DisplayModeChanged
D) TextChanged
3. Tarih seçici(DatePicker) kontrolünün takvimi açıldığı anda aşağıdaki olaylardan hangisi çalışır?
A) ClandarClosed
B) CalendarOpened
C) DateChanged
D) DatePickerOpened
4. Onay kutusu(CheckBox) kontrolü üzerinde fare tıklatıldığında çalışan olay aşağıdakilerden hangisidir?
A) Click
B) Checked
C) UnChecked
D) MouseEnter
5. Radyo düğmesi(RadioButton) kontrolünün işaretli olma durumunu temsil eden özelliği aşağıdakilerden hangisidir?
A) Content
B) B)GroupName
C) C)IsChecked
D) D)IsEnabled

6. Aşağıdakilerden hangisi içerisinde yer alan kontrolleri açılır bir pencere içinde gösteren kontroldür?
A) Button
B) CheckBox
C) RadioButton
D) Popup
7. ItemsControl liste elemanlarının tutulduğu dizi aşağıdakilerden hangisidir?
A) Items
B) Data
C) DataContext
D) Content
8. DataGrid içerisinde sütun tanımlamaları aşağıdaki yapılardan hangisinde bulunur?
A) Rows
B) Cells
C) Columns
D) DataTemplate
9. ListBox kontrolünde seçili durumda bulunan elemanı temsil eden özelliği aşağıdakilerden hangisidir?
A) SelectedIndex
B) B)Items
C) C)SelectedItem
D) D)ItemsSource
10. Ses ve video dosyalarını sayfa üzerinde yürütmemizi sağlayan kontrol aşağıdakilerden hangisidir?
A) ScrollViewer
B) Border
C) DataGrid
D) MediaElement
11. MediaElement kontrolünün video içeriğini yürütmesi sırasında zaman ilerlemesinin değerini tutan özelliği aşağıdakilerden hangisidir?
A) BufferingProgress
B) Position
C) Balance
D) Stretch

12. MediaElement kontrolünün video içeriğini otomatik olarak oynatmaya başlamasını sağlayan özelliği aşağıdakilerden hangisidir?
A) AutoPlay
B) Balance
C) Volume
D) Stretch
13. Veri bağlama(Binding) işleminde kullanılan veri aktarım yönlerinden hangisi sadece kaynak nesnenin özelliği değiştirildiğinde hedef nesnenin özelliğinin değişmesini sağlar?
A) OneWayToSource
B) OneWay
C) TwoWay
D) OneTime
14. Veri bağlama(Binding) işleminde kullanılan veri aktarım yönlerinden hangisi hedef ve kaynak nesnelerin özelliklerinin birbirine bağlı olarak değişmesini sağlar?
A) OneWayToSource
B) OneWay
C) TwoWay
D) OneTime
15. Veri bağlama(Binding) işleminde kullanılan veri aktarım yönlerinden hangisi hedef nesnenin özelliğinin değerinin bir defaya mahsus olarak kaynak nesnenin özelliğinden alınmasını sağlar?
A) OneWayToSource
B) OneWay
C) TwoWay
D) OneTime

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	A
2	B
3	C
4	D
5	Value
6	Is Direction Reserved

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	A
2	B
3	D
4	C
5	A
6	B
7	IsOpen
8	HorizontalScrollBar Visibility

ÖĞRENME FAALİYETİ-3'ÜN CEVAP ANAHTARI

1	A
2	B
3	C
4	D
5	A
6	B

ÖĞRENME FAALİYETİ-4'ÜN CEVAP ANAHTARI

1	C
2	B
3	B
4	C
5	Volume
6	IsMuted

ÖĞRENME FAALİYETİ-5'İN CEVAP ANAHTARI

1	Veri Bağlama
2	Binding
3	Path
4	Mode
5	DataContext
6	DataTemplate

MODÜL DEĞERLENDİRME'NİN CEVAP ANAHTARI

1	C
2	A
3	B
4	A
5	C
6	D
7	A
8	C
9	C
10	D
11	B
12	A
13	B
14	C
15	D

KAYNAKÇA

- MACDONALD Matthew, **Pro Silverlight 5 in C#, Apress**, Amerika, 2012.
- GÜLER Erman, **Silverlight 3**, Pusula Yayıncılık, İstanbul, 2010.